BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

MODELING AND SIMULATION OF SPANNING-TREE PROTOCOL

MODELOVÁNÍ A SIMULACE SPANNING-TREE PROTOKOLŮ

TERM PROJECT
SEMESTRÁLNÍ PROJEKT

AUTHOR **Bc. SIMONA POLÁČEKOVÁ**
AUTOR PRÁCE

SUPERVISOR **Ing. VLADIMÍR VESELÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání diplomové práce

Studentka: **Poláčeková Simona, Bc.**
Program: Informační technologie
Obor: Počítačové sítě a komunikace
Název: **Modelování a simulace spanning-tree protokolů**
Modeling and Simulation of Spanning-Tree Protocol
Kategorie: Počítačové sítě
Zadání:

1. Analyzujte protokoly RSTP a MSTP a prostudujte jejich chování na Cisco zařízeních.
2. Zhodnoťte stav současných implementace protokolů STP a RSTP pro OMNeT++.
3. Dle doporučení vedoucího implementujte primárně podporu MSTP (a sekundárně případná vylepšení RSTP) do frameworku ANSAINET/INET4+ v prostředí OMNeT++.
4. Ověřte chování implementovaných simulačních modelu vůči reálné topologii a analyzujte výsledky.

Literatura:

- Wehrle, Klaus, Mesut Günes, and James Gross, eds. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.
- Kraus, Z.: *Modelování a analýza spolehlivosti počítačové sítě VUT,* diplomová práce, Brno, FIT VUT v Brně, 2011.
- ANSI/IEEE 802.1Q-2005, Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Veselý Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 27. října 2020

Abstract

This term project deals with the functionality of Spanning Tree protocols, especially the Rapid Spanning Tree Protocol, and the Multiple Spanning Tree Protocol. The primary usage of spanning tree protocols is the prevention of loops within the data link layer, the prevention of a broadcast storm, and also deals with redundancy in the network. Moreover, the project contains the description of configuration of these protocols on Cisco devices. The main goal of this thesis is to implement the Multiple Spanning Tree protocol into INET framework within the OMNeT++ simulation system. Then, the implemented solution is tested and it's functionality is compared with the real behaviour in a Cisco network.

Abstrakt

Táto diplomová práca sa zaoberá funkcionalitou Spanning Tree protokolov, predovšetkým pre protokoly Rapid Spanning Tree Protocol a Multiple Spanning Tree Protocol. Spanning tree protokoly sa predovšetkým používajú pre prevenciu slučiek v rámci vrstvy dátového spoja, prevenciu broadcast storm a tiež riešia redundanciu v sieti. Ďalej je v tejto práci zahrnutý opis konfigurácie týchto protokolov na Cisco zariadeniach. Hlavným cieľom práce je implementácia Multiple Spanning Tree protokolu do prostredia INET vrámci OMNeT++ simulačného systému. Implementované riešenie je následne testované a funkcionalita je porovnávaná voči reálnemu chovaniu v Cisco sieti.

Keywords

switching, STP, Spanning Tree Protocol, RSTP, Rapid Spanning Tree Protocol, MSTP, Multiple Spanning Tree Protocol, OMNET++, ANSAINET, INET4+, Network modeling and simulation

Kľúčové slová

prepínanie, STP, Spanning Tree Protocol, RSTP, Rapid Spanning Tree Protocol, MSTP, Multiple Spanning Tree Protocol, OMNET++, ANSAINET, INET4+, modelovanie a simulácia sietí

Reference

POLÁČEKOVÁ, Simona. *Modeling and Simulation of Spanning-Tree Protocol.* Brno, 2021. Term project. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vladimír Veselý, Ph.D.

Rozšírený abstrakt

V tejto diplomovej práci sa zaoberám všeobecnou funkcionalitou, implementáciou a testovaním Spanning Tree protokolov. Tieto protokoly sa zameriavajú na tri hlavné úlohy: odstránenie nekonečných slučiek, zabránenie broadcast storms a blokovanie redundantných ciest. Okrem toho poskytuje STP aj možnosť obnovenia spojov, medzi ktorými bola prerušená aktívna cesta.

Spanning Tree algoritmus vo všeobecnosti funguje na voľbe rol pre switche a stavov na ich portoch. Pre zistenie a aktualizáciu týchto informácií si switche pravidelne posielajú BPDU správy, ktoré obsahujú aktuálne hodnoty pre všetky parametre na switchy. Zmeny v aktívnej topológii sú taktiež rozoznané formou BPDU správ.

Poznáme viaceré typy Spanning Tree protokolov. Prvým bol samotný STP, založený na IEEE štandarde 802.1D. Jeho nástupcom je Rapid Spanning Tree protokol, ktorý poskytuje rýchlejší prechod portu do stavu "forwarding" a taktiež rýchlejšie zotavenie po vypadnutí aktívnej cesty.

Ďalším variantom STP protokolu je MSTP (Multiple STP). Tento protokol poskytuje pre switche viacero ciest na posielanie dát medzi VLAN so zameraním na ich vyváženie. MSTP zadeľuje switche do regiónov. Aby dva switche patrili do rovnakého regiónu, musia mať rovnaký MST konfiguračný identifikátor, ktorý pozostáva zo štyroch častí: Configuration Identifier Format Selector, Configuration Name, Revision Level a Configuration Digest. Posledný spomenutý parameter je vytvorený na základe MST konfiguračnej tabuľky. Táto tabuľka obsahuje mapovanie VLAN na MST inštancie. V rámci regiónu môže existovať viacero MST inštancií. MSTP prepája tieto regióny do jediného stromu – Common and Internal Spanning Tree. Z hľadiska celého stromu sa regióny javia ako samostatné switche.

Hlavným cieľom tejto práce je implementácia Spanning Tree protokolov do prostredia INET v rámci OMNeT++ simulačného systému. OMNeT++ je rozšíriteľný framework používaný primárne na tvorbu sieťových simulátorov. Je založený na komponentoch a obsahuje integrované prostredie pre vývojárov odvodené z Eclipse s dodatočnou funkcionalitou, napr. na tvorbu a konfiguráciu sieťových modelov a analýzu výsledkov simulácií. INET je rozšírením OMNeT++ o sieťové modely pre rôzne protokoly ako napríklad Ethernet, IPv4, TCP a iné. Keďže STP a RSTP protokoly už sú súčasťou INET frameworku, hlavným zameraním tejto práce je vytvorenie MSTP.

MSTP modul bude bežať v rámci zloženého modulu EthernetSwitch. Vďaka podobnosti MSTP s inými protokolmi, ktoré už sú súčasťou INET frameworku, je možné pri implementácii použiť už existujúce moduly, ktoré budú rozšírené o MSTP funkcionalitu. Medzi tieto moduly patrí L2NetworkConfigurator modul, vylepšený o možnosť konfigurácie MSTP vlastnostní pre porty.

Konfigurácia MSTP modulu zahŕňa nasledujúce parametre: forwardDelay, maxAge, txHoldCount, maxHops, bridgePriority, mstConfig a mstiBPConfig. Posledné dva parametre reprezentujú XML konfiguráciu pre mapovanie MST inštancií na VLAN a pre nastavenie priorít switchov na jednotlivých inštanciách. Činnosť MSTP je implementovaná prostredníctvom stavových automatov na základe štandardu IEEE 802.1Q (2018).

Na účely testovania bola navrhnutá topológia, ktorá obsahuje sedem switchov rozdelených do dvoch regiónov a jeden ďalší switch bez príslušnosti k regiónu (switch predstavuje vlastný región). Jednotlivé parametre na portoch a switchoch boli nastavené tak, aby bolo možné otestovať čo najviac funkcionalít protokolu. V rámci regiónov boli nakonfigurované inštancie. Niektoré switche mali zmenenú hodnotu priority tak, aby sa stali rootom pre

danú inštanciu. Na zmenu predvolenej cesty k root switchu sa použil parameter ceny pre port.

Počas testovania boli overené aj zmeny v topológii v prípade výpadku linky a jej obnovenia. Tieto zmeny boli porovnávané s chovaním na Cisco switchoch. Správanie protokolov bolo v týchto prípadoch odlišné, pretože Cisco implementácia zahŕňa funkcionalitu na detekciu jednosmerného výpadku spojenia, ktoré však nie je súčasťou žiadneho IEEE MSTP štandardu. Implementácia pre INET neobsahuje toto vylepšenie.

Táto diplomová práca môže byť prínosom pre používateľov INET frameworku so záujmom simulovať sieť s MSTP protokolom. V budúcnosti môže byť riešenie doplnené o prepojenie MSTP modulu s ďalšími modulmi vrstvy dátového spoja, predovšetkým s modulmi pre VLAN. Okrem toho môže byť pridaná kompatibilita s predošlými verziami STP protokolu (STP, RSTP). Práca by mala byť v blízkej budúcnosti začlenená do oficiálneho INET frameworku.

Modeling and Simulation of Spanning-Tree Protocol

Declaration

I hereby declare that this master's thesis was prepared as an original work by the author under the supervision of Ing. Vladimír Veselý, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Simona Poláčeková
May 17, 2021

Acknowledgements

Firstly, I would like to thank my supervisor Ing. Vladimír Veselý, Ph.D for the chance to write this thesis under his supervision and for his guidance all along.

Additionally, many thanks to my fiancé Patrik for all his support, care, and for massages after hours of sitting in front of the laptop.

Last but not least, thanks to my parents Eva and Ľubomír and siblings Michaela and Tomáš for always supporting me throughout the entire time.

In gratitude, I would like to share my favorite recipe for tacos in a Swedish way. I first tasted it after my brother's Erasmus studies in Sweden, therefore again, additional thanks to my brother.

We will need the following ingredients for this recipe: 500g of mixed pork and beef ground meat, tacos seasoning, tortillas, grated cheese, sour cream, tortilla chips and vegetables according to your taste, I prefer corn, avocado, tomato, cucumber, and salad. Someone may also like to add a chopped red onion.

Firstly, we roast the meat until it changes color, then we add the seasonings and braise for about 15 minutes. In the meanwhile, we chop all the vegetables and prepare other ingredients. My best practice is to put them into separate bowls so that everything is ready. Once the meat is fully cooked, heat the tortillas in the microwave. Now we take the following steps: take one tortilla, spread the sour cream on one half of the tortilla, add meat, vegetables, cheese, and sprinkle with crushed tortilla chips for a crispy taste. We fold the tortilla, first the sides and then the bottom. Enjoy!

Contents

Chapter 1

Introduction

1.1 Motivation and Goals

In the past few years, the world of networking has changed tremendously. More and more technologies and protocols variations are used in this sphere. Therefore, it is needed to have a possibility to test and optimize those technologies and protocols before their actual application to practice. It is also a requisite for their refinement. Modeling and simulation are a good option for that purpose. With this option, we can also prevent complications during deployment of changes to a networking configuration and test the suitability of the changes subsequently.

Spanning tree protocols are operating at the second layer of the OSI model of computer networking, the data link layer. The primary usage of spanning tree protocols is the prevention of the layer 2 loops, broadcast storms, and it also deals with redundancy in the network.

This thesis deals with modeling and simulation of spanning tree protocols. The Multiple Spanning Tree Protocol (MSTP) is not a part of the INET implementation yet. The main purpose of the thesis is to implement a model of MSTP into an INET4++ interface within an OMNeT++ simulation system, test and evaluate the results of the solution afterwards.

1.2 Thesis Structure

In Chapter 2, we will look into the spanning tree protocols. It contains a description of a basic Spanning Tree Protocol (STP), but also its successors, the Rapid Spanning Tree Protocol (RSTP), and the Multiple Spanning Tree Protocol (MSTP). The information about the configuration of these protocols on Cisco devices is encapsulated in Chapter 3.

Chapter 4 describes the implementation and the technologies used during the development. These are the discrete event simulator OMNeT++, INET framework and ANSA, which is an extension of the INET framework. In this chapter, we additionally look into existing solutions for implementations of STP protocols in the OMNeT++.

In Chapter 5, we focus on the conducted tests and elaboration of the implemented Multiple Spanning Tree Protocol. We compare the solution with a referential behavior on Cisco switches.

The last Chapter 6 includes the evaluation of the progress within this term project and the plans for the future work on the diploma thesis.

Chapter 2

Spanning Tree Protocols

To protect the network from a failure such as a breakdown of a switch or a damaged network cable, redundant devices and links are established into the network. However, this solution causes loops on a network which inflicts frames duplication and even worse broadcast storms. The Spanning Tree Protocol was developed to deal with these difficulties. It was originally created for bridges, but nowadays it is used also for switched LAN topologies. In the following sections we will talk about more recent switched networks.

2.1 Spanning Tree Algorithm and Protocol

The main role of STP algorithm and protocol is to reduce topology (Bridged LAN topology, Switched LAN topology) to a single logical tree – the spanning tree. The resulting spanning tree ensures that the following conditions are met [16]:

- eliminated data loops – no more than one active path between any two end stations,
- automatic reconfiguration of the spanning tree topology – in case of a failure, a breakdown or an addition of a new switch to the topology.

2.1.1 STP Operations

For the spanning tree algorithm to work, the root bridge must be elected at first. The election process is under way on the basis of sending configuration BPDUs (Bridged Protocol Data Units). After the switches are turned on, they start to send the BPDUs to its neighbours and they behave as the root bridge themselves at first.

The BPDU contains information about the switch. An important field in the BPDU for the election process of the root bridge is the Bridge Identifier (BID). It consists of two parts:

- a Bridge Priority (default value 32,768),
- and a MAC address of the switch.

You can see a more specific BPDU structure and its description in Section 2.1.5. The root bridge is elected based on the lowest (i.e., superior) value of the BID. If the Bridge Priority was not modified within manual configuration, then the lowest MAC address represents the root bridge. When a switch receives a BPDU with a lower BID, it hands over these newly

determined BPDU to its neighbours instead of their own BPDU [25]. Finally, only the root bridge BPDU is advertised.

After the root bridge is found out, the next step is to identify the root ports for every switch which is not the root bridge. The way to do this is to calculate the least-cost path to the root bridge. The cost of the path (STP Cost, see in Section 2.1.6) depends on the speed of the interface on a network segment between switch and the root bridge. If the cost is the same for two or more paths, the next factor to determine the root port is the lowest sender BID. When also the BIDs are the same, the Port Identifier field in the BPDU of the sending switch is used. The Port Identifier is structured similarly to a Bridge Identifier and is comprised of:

- a Port Priority (default value 128) – may be configured,
- a unique port number.

Other port roles 2.1.2 and states 2.1.3 are derived with consideration to the previously mentioned determination of root ports for all non-root bridges in the topology. After acquiring the whole topology, the Spanning Tree Protocol is ready to operate.

With BPDU exchange mechanism, each bridge establishes own MAC address table, also called CAM table (Content Addressable Memory table), which is used to forward frames in the topology. On a typical Cisco switch, each MAC address entry has a default ageing time of five minutes (300 seconds) [20], which means that only after five minutes of a host inactivity is its entry removed from the MAC address table. This can lead to a traffic loss when a topology change occurs in the active topology.

Behavior After Topology Changes

A topology change may occur because of a failure of a switch, a link or a port in a topology. The active topology is maintained by exchanging the BPDU messages. The failure is recognized when a port does not receive a BPDU within expected period.

To prevent a traffic loss, the information about topology changes must be spread to the whole STP network. The process of exchanging the Topology Change Notification BPDUs after a topology change goes as follows [6]:

- When a topology change appears on a non-root bridge, it sends the Topology Change Notification BPDU out of the Root Port forwarding to the root bridge.
- If this notification arrives to the designated port of a non-root bridge from the downstream bridge, it sends it out of its root port as is towards the root bridge.
- After the previous step, the non-root bridge sends a Configuration BPDU with the Topology Change Acknowledgement flag set towards the bridge from which the notification originated, thus informing the downstream bridge about successful receipt of the Topology Change Notification BPDU. It is done to prevent a replications of the BPDUs.
- When the notification arrives to the root bridge, the Configuration BPDU with the Topology Change Acknowledgement (TCA) flag and the Topology Change (TC) flag set is sent by the root bridge to the bridge from which it received the TCN BPDU at first. This device then distributes the Configuration BPDU with only the TC flag set further. Then the root bridge generates the Configuration BPDU with only the TC

flag set and sends it to all other bridges and it is distributed to the whole spanning tree topology. Those BPDUs are sent for the duration of 35 seconds, the sum of the Maximum Age Timer and the Forwarding Delay Timer. The TC flag informs about a change in the topology.

- After receiving the Configuration BPDU with TC flag set, bridge sets the aging time for each entry of its MAC address table to the Forwarding Delay Timer, which is 15 seconds. After this timer, the inactive MAC addresses are deleted and thus the convergence speed is increased.

When a topology change occurs, some port roles in the spanning tree topology may be changed to reestablish a new active topology. A recalculation of the whole spanning tree may result from adding a new bridge with a smaller Bridge Identifier than the current root bridge, or from a failure of the root bridge.

2.1.2 Port Roles

A port role is determined for each bridge within the spanning topology in the Spanning Tree Protocol. The old version of Spanning Tree Protocol contained these roles: Root Port, Designated Port and Blocking Port. Since the the IEEE Standard 802.1D, version 1998 [16] was replaced and the STP has a successor – the Rapid Spanning Tree Protocol, the port roles are more specified in Section 2.2.2. Nowadays, same port roles may be applied for the STP.

2.1.3 Port States

The Spanning Tree Protocol functionality is provided by port states. Each port has assigned state in which it operates. Port states with their operations are summarized in the table 2.1. Following port states with their functionality are considered for STP [16]:

- Forwarding – a port in the forwarding state transmits and receives frames. Also a mac learning process is enabled. It process received BPDUs and submits BPDUs for transmission.
- Learning – sending and receiving frames is disabled to avoid temporary loops and frame duplication. Since in this state, the port is preparing to transfer to forwarding state, the mac learning process is enabled, so the number of unnecessarily relayed frames is reduced. Sending and receiving BPDUs takes place on this port too.
- Listening – listening port may still send and receive BPDUs. The learning process is disabled because of temporary loops prevention.
- Blocking – this type of port exists for preventing frame duplication. A port in blocking state only receives BPDUs which are processed afterwards.
- Disabled – in this state, the port is shut down administratively and does not participate in the active spanning tree topology.

Port state	Send & Receive Frames	Mac Learning	Send BPDUs	Receive BPDUs
Forwarding	✓	✓	✓	✓
Learning	✗	✓	✓	✓
Listening	✗	✗	✓	✓
Blocking	✗	✗	✗	✓
Disabled	✗	✗	✗	✗

Table 2.1: Operation of Port States, [27]

2.1.4 STP Timers

The STP process is based on exchanging BPDUs. To determine errors in the network or to maintain the spanning tree it is also important to have timers for those BPDUs. The following default timers exists for STP [30]:

- Hello timer – the default value of this timer is two seconds. This timer determines how often is the BPDU periodically sent from the bridge. When bridge does not receive any BPDU from the root bridge, it stops to send the BPDUs. The purpose is to find out possible failure of a link.
- Forward Delay timer – represents a delay period for STP root ports and designated ports before transitioning from the listening and learning state to the forwarding state. This type of timers ensures there is sufficient time to determine and eliminate loops. Default value for this timer is 15 seconds. To transmit from both listening and learning states it results in 30 second delay duration.
- Max age timer – specifies maximum possible age for the BPDU. The BPDU is forwarded when its Message Age value is smaller or equal to the Max Age value, otherwise the configuration BPDU is discarded. The default value is 20 seconds (10-times the Hello Timer).

The following relationships between timers should be observed to maintain interoperability in the network:

$$2 * (Forward\ Delay\ timer - 1.0\ seconds) \geq Max\ Age\ timer$$

$$Max\ Age\ timer \geq 2 * (Hello\ timer + 1.0\ seconds)$$

2.1.5 Bridged Protocol Data Unit

The Bridged Protocol Data Unit (BPDU) structure is displayed in Figure 2.1. The BPDUs are sent from a non-root bridge every hello-time only while it receives BPDUs on its root port. The BPDU contains the following fields [8]:

- Protocol Identifier – fixed value 0x0000, identifies the STP family protocols.
- Protocol Version Identifier – fixed value 0x00 for STP.
- BPDU Type – represents type of the BPDU. The configuration BPDU has value 0x00. The Topology Change Notification (TCN) BPDU has value 0x80. We use a configuration BPDU to select the root bridge, root ports and designated ports. TCN

is used in case of a port transition to a forwarding state or when one of the forwarding and learning ports transitions to blocking or disabled state. In case of TCN only first 4 bytes are considered.

- Flags – the purpose of BPDU is indicated by this 8-bit field. The lowest bit represents the Topology Change flag (TC), the highest represents the Topology Change Acknowledge flag (TCA). Other bits are unused.
- Root Identifier – contains the priority number and MAC address of the root bridge.
- Root Path Cost – represents the cost of the path from the sender to the root bridge.
- Bridge Identifier – contains the priority number and MAC address of the sender.
- Port Identifier – identifies the port from which a sender forwards the configuration message to the network; contains the priority number and global port number of the port.
- Message Age – age of the current configuration BPDU while spread out the network.
- Max Age – indicates the maximum possible age for the configuration BPDU. After passing this timer, the configuration BPDU shall be deleted.
- Hello Time – a periodic transmission interval for configuration BPDU. How often the BPDU is send from the bridge.
- Forward Delay – represents waiting time that bridge should wait before a port state can change.

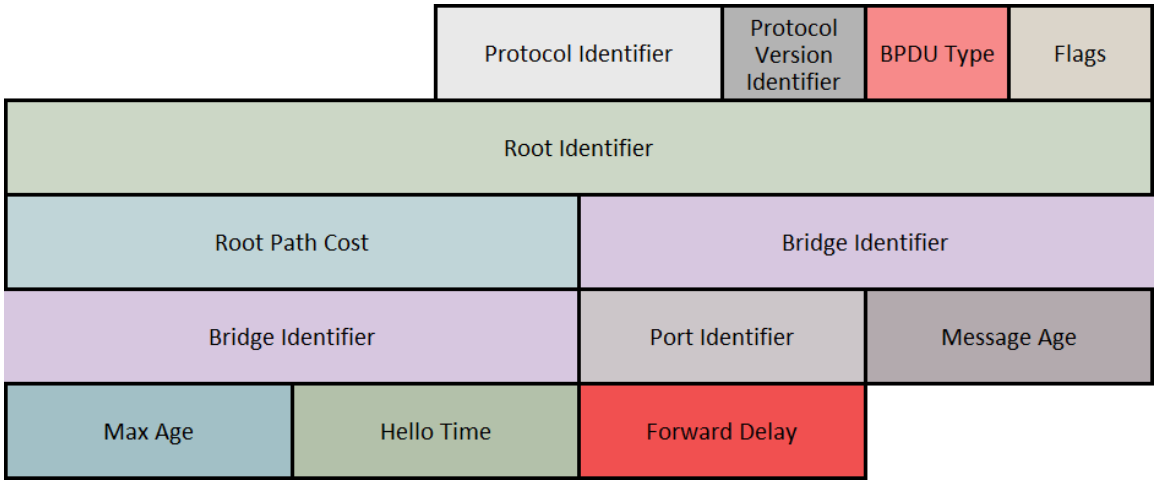

Figure 2.1: Bridged Protocol Data Unit structure [16].

The least conditions that must be met for processing the BPDU by the Spanning Tree Protocol:

- the BPDU consists at least of four bytes
- the Protocol Identifier value is 0x0000
- the BPDU Type stands for a Configuration BPDU with Message Age parameter lesser than the Max Age parameter or it stand for a Topology Change Notification BPDU.

2.1.6 Spanning Tree Costs

Path cost from a non-root switch to the root bridge depends on a speed of a networking technology. STP costs are associated to speeds of an interface on a network segment, see in Table 2.2.

Link Speed	Recommended Cost Value	Recommended Cost Range	Cost Range
4 Mb/s	250	100 – 1000	1 – 65535
10 Mb/s	100	50 – 600	1 – 65535
16 Mb/s	62	40 – 400	1 – 65535
100 Mb/s	19	10 – 60	1 – 65535
1 Gb/s	4	3 – 10	1 – 65535
10 Gb/s	2	1 – 5	1 – 65535

Table 2.2: STP – Path Costs associated to Link Speeds, [16]

2.2 Rapid Spanning Tree Protocol

The Rapid Spanning Tree Protocol is a replacement for the previously mentioned Spanning Tree Protocol. It is defined in the standard IEEE 802.1D, 2004 Edition [15]. The RSTP now represents the default Spanning Tree Protocol and the main purpose of this protocol remains the same as in the STP.

The protocol provides faster (therefore, the „rapid“ in the name) transition of a port to the forwarding state. It also provides faster recovery from a failure by including the port roles in the computation of port states and by allowing explicit acknowledge signals indicating a port wants to enter the forwarding state [26].

2.2.1 RSTP Operations

The process of selection of the root bridge remains the same as in the spanning tree algorithm. The root port is also chosen similarly as in the spanning tree algorithm, but the recommended values for the least cost paths differ. The recommended least cost paths for the Rapid Spanning Tree Protocol are mentioned below in Section 2.2.5. The determination of other port roles is described in Section 2.2.2.

The Rapid Spanning Tree Protocol ensures a rapid transition to forwarding state, which is the most important feature in comparison with the STP. The following concepts are considered in achieving this [29]:

- Edge Ports – a port that is directly connected to end stations and is not connected to any other bridge device. There cannot be a loop upon topology changes caused by the edge port. When the edge port parameter is set, the port transitions to the forwarding state forthwith, skipping the listening and learning states. After the configuration BPDU is received on the edge port, the edge port attribute is lost. The port becomes a non-edge port and the spanning tree recalculation is performed. This causes a network flapping, which can be prevented with BPDU protection. It corresponds to the Cisco PortFast feature[1].

[1]The description of the PortFast – https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4000/8-2glx/configuration/guide/stp_enha.html#wp1046787

- Link Type – the duplex mode of a port automatically decides about the link type. If a port is operating in the half-duplex mode, it is considered as a shared port – on a link between a switch and a hub. In the full-duplex mode, the port is considered a point-to-point link – links two switches. The full-duplex point-to-point links may be eventually rapidly transitioned to the forwarding state, see the Proposal/Agreement Process further in this section.

Proposal/Agreement Process

After election of the designated port in the RSTP, the port firstly enters the Discarding state and afterwards it rapidly transitions to the Forwarding state thanks to the Proposal/Agreement (P/A) mechanism. This is considered a speed up according to the STP, where the port must wait for a two forward delay timers, transitioning firstly to the learning state and secondly to the forwarding state.

The P/A mechanism may be used only on point-to-point full-duplex links. It uses Proposal and Agreement messages to achieve the rapid transition to the forwarding state. When there is no agreement response to the proposal message on the designated discarding port, the port transitions to the forwarding state the same way as in the STP, after two forward delay timers. The P/A process does not rely on any timers and it is considered a fast mechanism to reestablish connectivity after a change in the active topology.

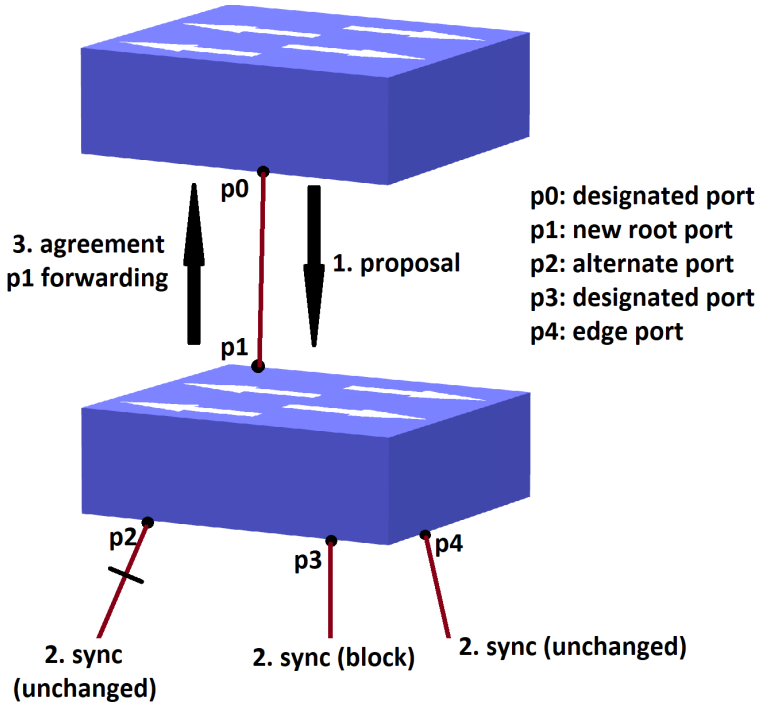

Figure 2.2: Rapid Spanning Tree Protocol – Proposal/Agreement Mechanism, part 1 [29].

We will describe the Proposal/Agreement process with a consideration to the topology displayed in Figure 2.2. Let us assume that we add a new point-to-point full-duplex link between switches named Root and A. The P/A process goes as follows [29]:

- Firstly, both switches act as a root bridge, therefore, both ports p0 and p1 are immediately considered the designated ports and send the configuration BPDUs.

- The switch A receives the better BPDU on the p1 interface, thus the p1 immediately knows that it is now the root port.
- The port p0 of the Root switch enters the Discarding state, so it sets the proposal bit in the sent BPDUs (step 1 in Figure 2.2). The proposal bit may be set only on a designated port which state is discarding or learning.
- After the switch A receives the BPDU with a proposal from the Root switch, it starts the sync to ensure that all its ports are in sync – set to one of the following, or blocked:
  - The port is in a discarding state, blocked.
  - The port represents an edge port.
- To show how the sync mechanism works, lets assume that the switch A has except the new root port the following ports:
  - an alternate port p2,
  - a designated forwarding port p3,
  - and an edge port p4.
- Port p2 is blocked with unchanged state. Port p4 is an edge port and it does not participate in the operation. Both ports meet one of the previously mentioned conditions to be in sync. Therefore, the switch A must change only the non-edge designated port p3, the port must be blocked and put in the discarding state. In this moment, all ports of the switch A are in sync (step 2 in Figure 2.2).
- Now the switch A can send the response BPDU with the agreement bit set back to the Root switch (step 3 in Figure 2.2). The BPDU carries same information as the proposal BPDU, except that instead of the proposal bit, the agreement bit is set.
- When the Root switch receives the corresponding agreement to its proposal on the port p0, the port can immediately transition to the forwarding state. We may see this step (step 4) in Figure 2.3.
- The process continues to the downstream bridges. We may see, the port p3 is in a designated discarding state after the sync. It sends the proposal to its neighbors in pursuit of transition to the forwarding state as soon as possible.

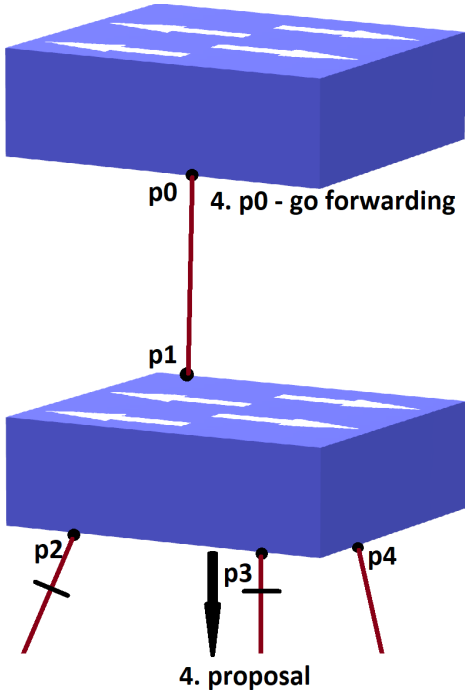

Figure 2.3: Rapid Spanning Tree Protocol – Proposal/Agreement Mechanism, part 2 [29].

RSTP Topology Change

In the Rapid Spanning Tree Protocol, the topology change is caused only when a non-edge port transitions to the forwarding state. A failure of a device and lost connectivity is unlike in the STP not considered as a topology change. So, a port which transitions to the blocking state does not generate a TC BPDU.

A propagation of a topology change to the whole topology does not need to be acquired through the root bridge anymore. When a topology change is detected by an RSTP bridge, the following actions occur [10]:

- The initiator bridge starts a TC While timer for its non-edge designated ports and also the root port, when necessary. The While timer is set to the two times of the hello time.
- It clears the MAC addresses related to all this ports mentioned in previous step.
- The outbound BPDUs have the TC bit set for the time of the TC While timer.

After the TC BPDU is received on a bridge from its neighbor, the following actions are performed on this bridge [10]:

- It clears the MAC addresses on all ports, except the port on which the TC BPDU was received.
- It starts a While timer on all its designated ports and root port and sends BPDUs with the TC bit set from them.

The Topology Change Notification is spread to the whole topology much faster as in the STP. The process does not need to notify the root bridge at first, which then maintains

the topology by informing all other bridges about a change for the time of the max age plus the forward delay.

2.2.2 RSTP Port Roles

Each port in the rapid spanning tree topology has an assigned role. The following types of ports are defined for Rapid Spanning Tree Protocol [23]:

- Root Port – one root port is elected on each non-root bridge. The root port represents the most optimal path (path with the lowest cost) to the root bridge. The election process is same as in the Spanning Tree Protocol, described in Section 2.1.1. This port type is the only one which communicates with the root bridge both ways – receives and forwards frames.
- Designated Port – a selection of a designated port is ongoing as follows: when one end of a link has a resolved root port, then the other end represents a designated port. Every link connection on a designated bridge has one designated port. Also all ports on the root bridge function as designated ports. A main role of this port is to forward traffic from the root bridge toward a non-root bridges.
- Alternate Port – selected only on a link with no root port. Does not participate in active topology. It is activated only when root port fails and then it provides alternate path to the root bridge (a backup port for the root port).
- Backup Port – a similar functionality to the alternate port, except it provides a backup for designated ports, in case of their failure. A backup port existence is possible only on bridges, where more than one link is connected to the same LAN – redundant links, and the bridge has a designated port on one of those links.
- Disabled Port – port is shut down and does not participate in the active spanning tree topology.

2.2.3 RSTP Port States

There are only 3 port states for the Rapid Spanning Tree Protocol in comparison with 5 port states for the previously mentioned STP. Port states listening, blocking and disabled were merged into one state – the discarding state. The port states help to control the functionality of the protocol and the operation of the Learning and Forwarding processes. All port states receive the BPDUs for processing. The characteristics of the RSTP port states [22]:

- Forwarding – ports in the forwarding state establish the active topology. The forwarding state may be only seen in a stable active topology. While synchronization and topology changes, forwarding of frames is only possible by a proposal and agreement process described in Section 2.2.1.
- Learning – the learning state accepts data frames in order to learn MAC addresses and to build the MAC address table. Used in stable active topology and while synchronization and topology changes.
- Discarding – the discarding state does not forward the data frames to the network, and thus it prevents the loops. Used for backup and alternate port roles. Same as the

learning state, used in stable active topology and while synchronization and topology changes.

2.2.4 Rapid Spanning Tree BPDUs

The rapid spanning tree BPDUs has some differences from the spanning tree BPDU. The RST BPDU is the second version of the BPDU protocol since the version 0 of STP BPDU. In the Rapid Spanning Tree Protocol, a bridge sends a BPDU every hello-time without any matter of receiving one on its root port. The structure for RST BPDU is displayed in Figure 2.4.

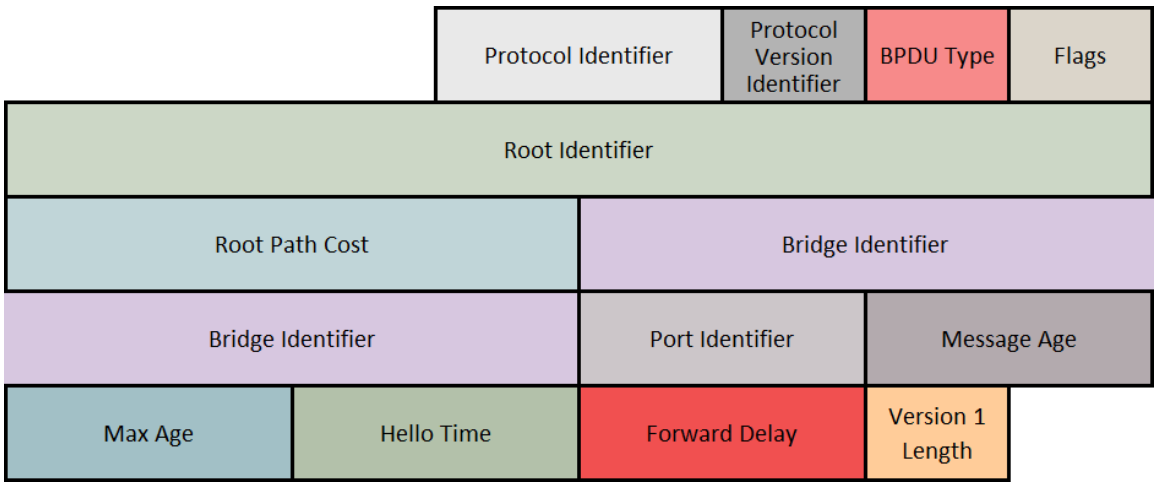

Figure 2.4: Rapid Spanning Tree Protocol – Bridged Protocol Data Unit structure [15].

The description of the RST BPDU's structure is following [15]:

- Protocol Identifier – fixed value 0x0000, same as STP.
- Protocol Version Identifier – fixed value 0x02 for RSTP.
- BPDU Type – represents type of the BPDU. The configuration BPDU has value 0x02 which denotes RSTP BPDU. The Topology Change Notification BPDU remains same as in the STP.
- Flags – the encoding for the RST BPDU flags may be seen in Figure 2.5. Following flags may be encoded: Topology Change flag, Proposal flag, Port Role, Learning flag, Forwarding flag, Agreement flag, Topology Change Acknowledgement flag.
- The following parameters remains same as in the STP 2.1.5: Root Identifier, Root Path Cost, Bridge Identifier, Port Identifier, Message Age, Max Age, Hello Time, Forward Delay.
- Version 1 Length – takes 1 byte. When the value is 0x00, then it indicates there are no information from previous versions of the BPDU protocol.

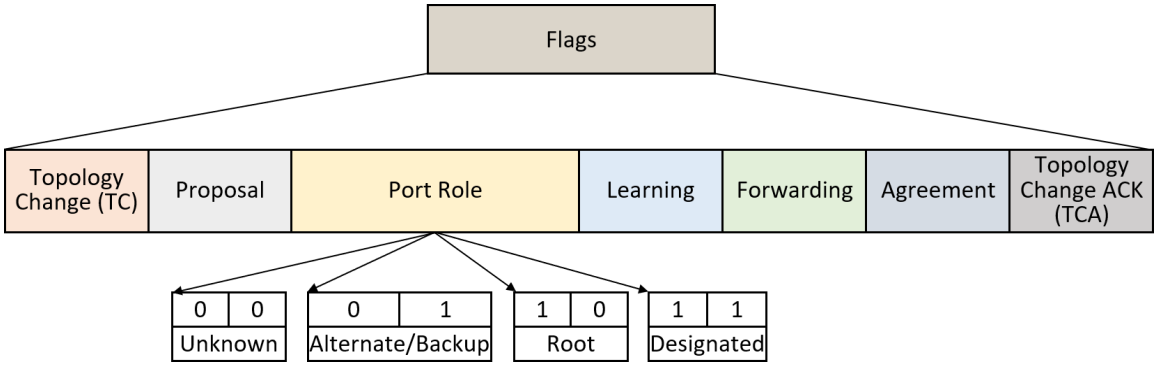

Figure 2.5: Structure of the Flags field in the Rapid Spanning Tree BPDU [15].

2.2.5 Rapid Spanning Tree Costs

Recommended values for the path costs from a non-root switch to the root bridge are showed in Table 2.3. The path cost parameter is associated to the speed of an interface, and may be set from range $1 - 200{,}000{,}000$ when supporting the 32 bit Path Cost values.

Link Speed	Recommended Cost Value	Recommended Cost Range
$\leq$ 100 Kb/s	200,000,000	20,000,000 – 200,000,000
1 Mb/s	20,000,000	2,000,000 – 200,000,000
10 Mb/s	2,000,000	200,000 – 20,000,000
100 Mb/s	200,000	20,000 – 2,000,000
1 Gb/s	20,000	2000 – 200,000
10 Gb/s	2000	200 – 20,000
100 Gb/s	200	20 – 2000
1 Tb/s	20	2 – 200
10 Tb/s	2	1 – 20

Table 2.3: RSTP – Recommended Path Costs associated to Link Speeds, [11].

2.3 Per-VLAN Spanning Tree

The basic STP does not consider Virtual Local Area Networks (VLANs). This may lead to the connectivity loss on the VLAN segments of the network. Possible solution is to create separate active topologies for each of VLANs. On each VLAN, a separate STP process is running. The network now considers VLANs and their ports, therefore, the link connecting the VLAN to the rest of the network may not be blocked, and the connectivity may not be lost for any VLAN. This approach is included within the per-VLAN Spanning Tree (PVST) Cisco proprietary protocol.

There is several variations for the PVST. They differ in version of STP (using STP or RSTP) and the communication on the links. The PVST variations are following [18]:

- PVST – this per-VLAN Spanning Tree uses the STP, and has a new Cisco proprietary enhancements[2]: PortFast, BPDU Guard, UplinkFast, BackboneFast, etc. With

[2]Detailed description of those enhancements may be seen at https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4000/8-2glx/configuration/guide/stp_enha.html

PVST, the communication on a link is performed on the basis of Cisco proprietary mechanism for VLAN encapsulation, the Inter Switch Link (ISL).

- PVST+ – the PVST+ uses the STP. The difference to the PVST is that PVST works only with ISL, which has a different frame format, and the PVST+ is compatible with IEEE 802.1Q encapsulation and frame format.
- Rapid-PVST – same as the PVST, except it uses the RSTP.
- Rapid-PVST+ – same as the PVST+, except it uses the RSTP.

2.4 Multiple Spanning Tree Protocol

In the previously mentioned versions of STP, except the PVST, the spanning tree is shared by all VLANs in the network. There is no load balancing of the data traffic, which causes bandwidth waste. The Multiple Spanning Tree Protocol, defined in the IEEE 802.1s standard [17] improves the aforementioned deficiencies. It provides multiple separate paths for data forwarding and implements load balancing of the frames between VLANs.

In this section, we will focus on the functionality of the MSTP. We will describe the Multiple Spanning Tree Instances (MSTI) within the Multiple Spanning Tree (MST) Regions, which are composed of LANs and MST Bridges, and how are these MST Regions connected within a single Common and Internal Spanning Tree (CIST). We will also describe the composition of MST BPDU and the differences in port roles for MSTP.

2.4.1 MST Configuration Identifier (MCID)

The MSTP specifies an MST Configuration Identifier (MCID). The identifier serves for propagation of a bridge configuration for allocation of frames with assigned VIDs to one of Multiple Spanning Tree Instances (MSTIs). The agreement on allocating of VIDs to specific MSTIs is fundamental to all bridges within an MST Region. A duplication of frames, or frame loss may occur when the consistency of the allocation is not ensured.

To assure that the VIDs are allocated to the same spanning trees for all bridges within the MST Region, the MST Configuration Identifiers are transmitted together with other information in the BPDUs. The following fields are included in the MST Configuration Identifier [17]:

- Configuration Identifier Format Selector – fixed value 0x00, indicating the use of the MST Configuration Identifier.
- Configuration Name – a string encoded in 32 bytes representing the name of the MST Region.
- Revision Level – an unsigned integer value encoded in 2 bytes (0-65535) that identifies MST Region.
- Configuration Digest – this field denotes a signature (based on HMAC-MD5 authentication method) created from an MST configuration table. The MST configuration table is constituted by maximum of 4096 successive 2-byte elements. The elements in the configuration table contains a mapping of values of VIDs (except of first and

last elements, which contains the value 0) to MSTIDs. The Configuration Digest signature is encoded in 16 bytes, generated by using the signature key which is a string specified in Table 2.4.

Parameter	Mandatory Value
Configuration Digest Signature Key	0x13AC06A62E47FD51F95D2BA243CD0346

Table 2.4: MST Configuration Identifier – Configuration Digest Signature Key [17].

2.4.2 MSTP Port Roles

MSTP assigns a port role to each spanning tree used within the whole topology. Firstly, the port roles are assigned to the CIST, and to the MSTIs afterwards. The CIST uses the same port roles as the RSTP 2.2.2. The Multiple Spanning Tree Instance uses additional port role – Master port. Port roles are assigned on the basis of the spanning tree port priority vectors. The following port roles are defined by the MSTP [13]:

- Root Port
- Designated Port
- Master Port – a newly defined port role, used in the MSTI. The master port, same as the root port or designated port forwards data frames. More detailed description is contained in Section 2.4.7.
- Alternate or Backup Ports – substitute for other ports in case of their failure.
- Disabled Port – a port role, which represents a port that is not enabled. It is not participating in the active topology and neither in any of operating spanning trees.

2.4.3 MSTP Port States

Port states for MSTP remains same as the port states defined for RSTP 2.2.3. Root ports, master ports and designated ports are transitioned rapidly to the forwarding state. Alternate and backup ports are transitioned to the discarding port state as rapidly as possible without the risk of data loops [14].

2.4.4 Spanning Tree Priority Vectors

To determine the active topology for any type of a spanning tree, the Configuration Messages are sent between devices in the topology. The exchanged information is called a spanning tree priority vector [17]. After receiving a priority vector, the port role is decided on the basis of comparing the received vector with priority vector to be transmitted. The decision is made in favour of the lower priority vector values. A description of spanning tree priority vectors for the MSTIs is further in Section 2.4.7 and for CIST in Section 2.4.6.

2.4.5 MST Region

The MSTP region consists of one or more switching devices, which use the same MSTIs 2.4.7. For two devices to belong in the same region, we must ensure that MSTP is running on both devices, and also the following characteristics must be the same [28]:

- devices share the same configuration name,
- have the same configuration revision level,
- and incorporates the same mapping of respective VLANs (VIDs) to spanning tree instances (MSTIDs) within every MST Region. The mapping is stored in the MST configuration table of length of 4096 elements.

With the Multiple Spanning Tree Protocol, it is possible to divide network devices and VLANs into multiple regions (more than one device in a region). Those MST regions then behave as single bridges. The MST regions are interconnected as „single bridges“ to the Common Spanning Tree (CST). Inside the regions the connection is ensured with the Internal Spanning Tree (IST) or Multiple Spanning Tree Instances (MSTIs). The maximum number of instances created within one MST region is 16 [5]. Both, the connection outside, and inside the MST regions together form one Common and Internal Spanning Tree (CIST).

2.4.6 Common and Internal Spanning Tree (CIST)

All the network devices in the topology are connected with a single Common and Internal Spanning Tree. The CIST represents the overall spanning tree for such a topology. It has a part contained within each MST region - the Internal Spanning Tree (IST), which has the regional root as its root bridge, and operates like an MSTI [13]. The IST is also referred as MST instance 0. The CIST connectivity between regions is also referred as the Common Spanning Tree (CST).

An example of how the MSTP interconnects MST Regions into one Common and Internal Spanning Tree is displayed in Figure 2.6. Each Region behaves as a single switch from a perspective of a CST.

CIST Port Roles

Every port, which is enabled for MST, has a port role assigned to it for each spanning tree. At first, port roles are assigned to the CIST. The determination of the CIST root bridge and the port roles is in comparison with the RSTP with extended priority vectors within MST Regions. The process of assigning the port roles to the CIST is following [17]:

- CIST Root Port – assigned to all bridges except the CIST root bridge. Port that is a source of the root priority vector and provides the minimum cost path to the CIST root for the bridge where it is located. If the bridge is not a regional root, the minimal path leads through the regional root.
- CIST Designated Port – each port with designated priority vector, which is derived from the root priority vector. It ensures the least cost path for the attached LAN through the bridge where it is located further to the CIST root.
- CIST Alternate Port – each non Root Port with port priority vector received from another bridge. May ensure connectivity in case of a failure in the active topology.
- CIST Backup Port – each port with port priority vector received from another port on this bridge. Same as the alternate port, it may ensure connectivity in case of a failure in the active topology.

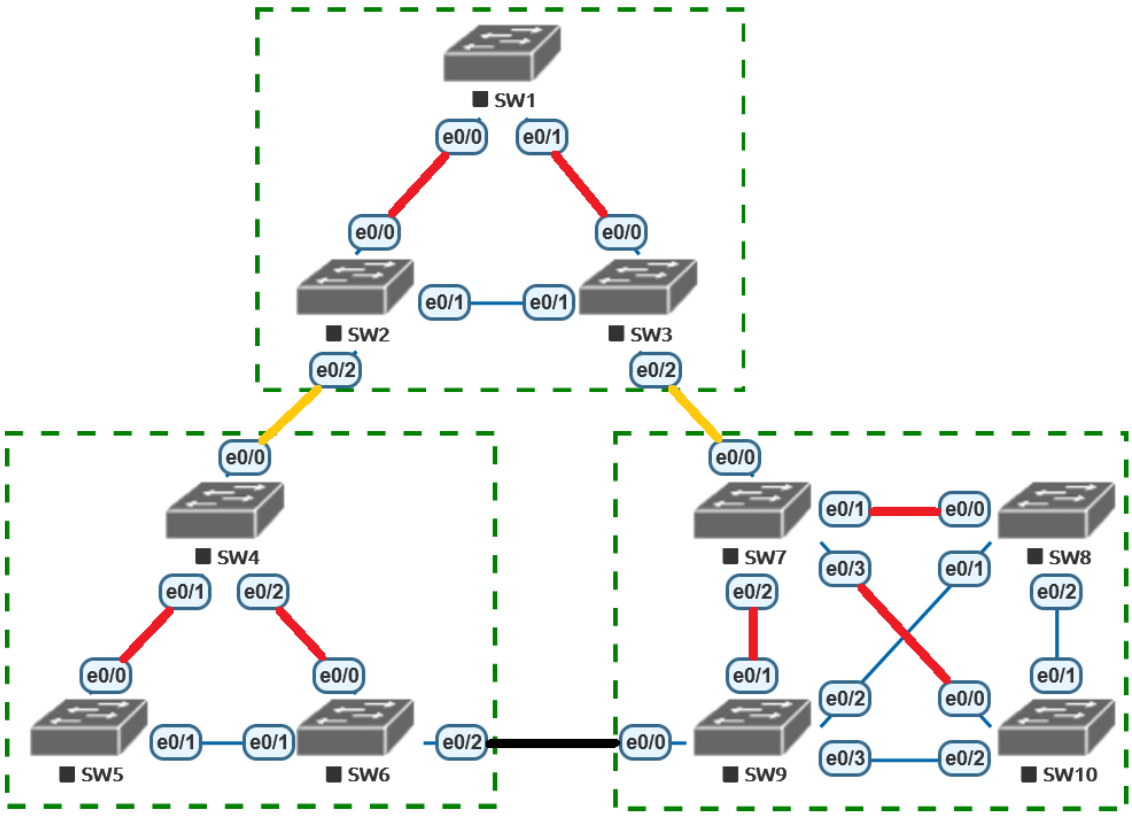

Figure 2.6: Common and Internal Spanning Tree (CIST) with three MST Regions. Reworked from [24, 2.2].

CIST Priority Vectors

The following components are contained within CIST priority vectors [13]:

- CIST Root Identifier – CIST root's Bridge Identifier.
- CIST External Root Path Cost – cost of a path between MST regions from the sending bridge to the CIST root.
- CIST Regional Root Identifier – stands for the Bridge Identifier for one of the following:
  - of the single bridge in a region whose CIST root port represents a boundary port.
  - of the CIST root if it is located within the region.
- CIST Internal Root Path Cost – the path cost to the CIST regional root.
- CIST Designated Bridge Identifier – the Bridge Identifier for the transmitting bridge for the CIST.
- CIST Designated Port Identifier – the Port Identifier for the transmitting bridge for the CIST.

- CIST Receiving Port Identifier – the Port Identifier is not present in configuration messages. It is used as a tiebreaker in case of equal priority vectors within a receiving bridge.

2.4.7 Multiple Spanning Tree Instance (MSTI)

The Multiple Spanning Tree Instance operates in similar way as the RSTP within an MST Region. The regional root identifier and internal path cost parameters are dedicated for MSTI. The process of selecting the regional root, and port roles computation uses modified priority vectors 2.4.4, which helps in constructing an active topology for MSTIs for every region.

MSTI Port Roles

In the MSTP, the port roles are assigned to all ports in the topology. The port roles are determined firstly for the CIST, and for the MSTI afterwards. In the MSTI, the following port roles may be set for its active topology within an MST Region and also at the region boundaries [17]:

- MSTI Master Port – ports with this role ensure a connectivity to a CIST root from the region where they are located to a CIST root outside this region. The MSTI master port is determined on the CIST root port which received the CIST port priority vector from an outside bridge. The CIST root port for the CIST regional root represents also the master port for all MSTIs.
- MSTI Root Port – this port role is determined for bridges that are not the MSTI regional root. It is a port which is a source of the MSTI priority vector. The MSTI root port provides the minimum cost path to the MSTI regional root.
- MSTI Designated Port – the MSTI designated port is every port which is endowed with the designated priority vector derived from the root priority vector. It ensures the least cost path for the LAN attached to it through the bridge to the regional root.
- MSTI Alternate Port – a port which does not represent a master port or a root port, and which contains a received:
  - port priority vector from another bridge,
  - or CIST port priority vector from bridge located in a different region.

  This port may provide connectivity in case of a failure in the active topology.
- MSTI Backup Port – the MSTI backup ports are all other ports that has a port priority vector received from another port on the bridge where they are located. This port may also provide connectivity in case of a failure in the active topology.

The MST instances within the region may contain different configuration parameters for each bridge or port, therefore the port roles may also vary. An example of instances is shown in Figure 2.7. Let us consider unchanged port cost parameters, and modified bridge priorities for the root switches in this example. The instance `MSTI0` is considered a CIST instance, so the port roles are determined as described in Section 2.4.6. For other MST instances, the port leading out of the Region towards the CIST root is considered Master

Port. The `MSTI2` instance with switch `SW9` as a MSTI regional root has all internal ports (ports that receive BPDUs from the same region) set to Designated Port Role. On other switches within the region, the ports leading to the MSTI regional root are the Root Ports. The rest of the ports in this example are Alternate Ports.

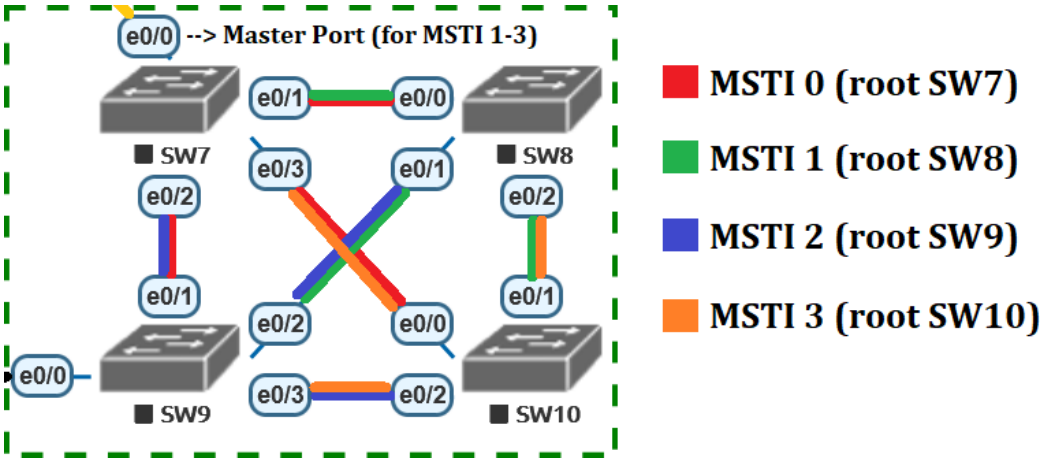

Figure 2.7: An example of Multiple Spanning Tree Instances within the MST Region. Reworked from [24, 2.3].

MSTI Priority Vectors

The MSTI priority vectors are only defined within an MST region, where they are totally and uniquely ordered. The following components are contained within MSTI priority vectors [13]:

- MSTI Regional Root Identifier – the Bridge Identifier of the MSTI regional root for the particular MSTI in which it is located, in this MST region.
- MSTI Internal Root Path Cost – a cost of the path to the MSTI regional root for the particular MSTI in which it is located, in this MST region.
- MSTI Designated Bridge Identifier – the Bridge Identifier for the transmitting bridge for this MSTI.
- MSTI Designated Port Identifier – the Port Identifier for the transmitting port for this MSTI.
- MSTI Receiving Port Identifier – the Port Identifier is not present in configuration messages.

2.4.8 Multiple Spanning Tree BPDUs

The Bridge Protocol Data Unit for MSTP is of version 3. The first thirteen fields of an MSTP BPDU does not differ from the previously mentioned RSTP BPDU 2.2.4. It consists of six more fields that are unique to MSTP. The MSTP BPDUs may be used as the Configuration Messages, the Topology Change Notification (TCN) Messages and besides that also as MSTI Configuration Messages.

An MST bridge supports 64 MSTIs at the most, which is also the limit for number of MSTI Configuration Messages configured in an MST BPDU [17]. The structure of MSTP BPDU is shown in Figure 2.8.

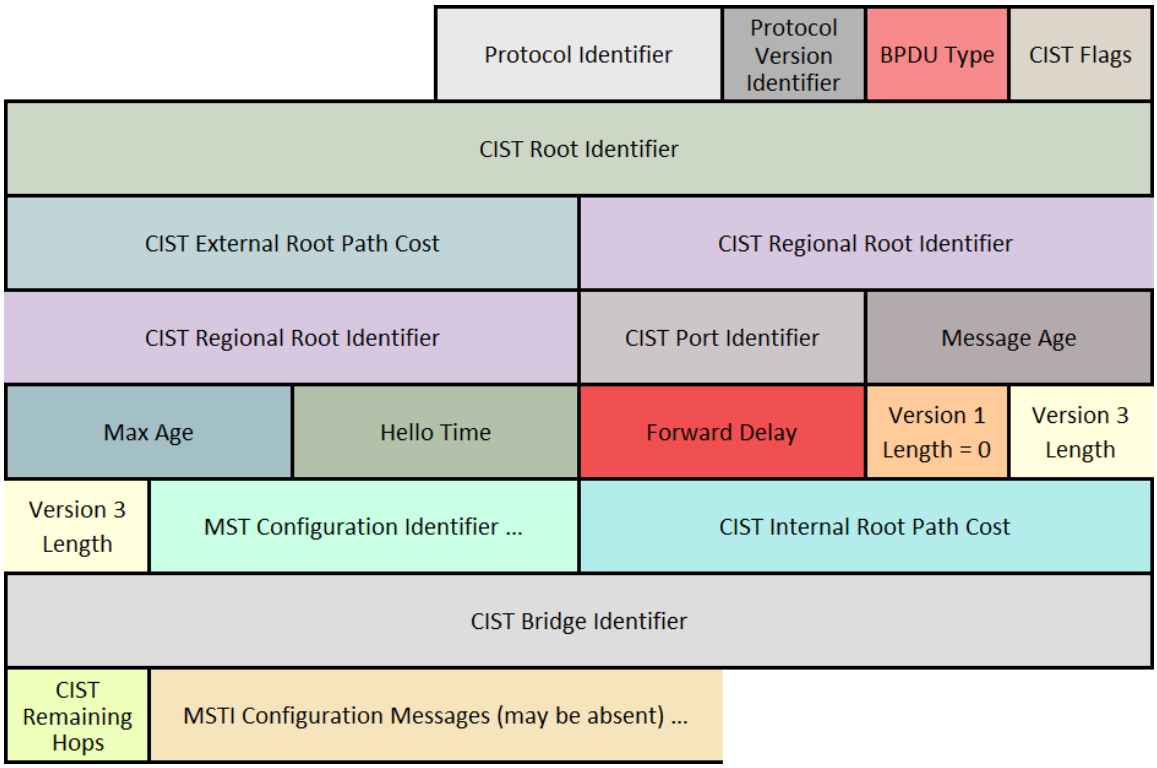

Figure 2.8: Multiple Spanning Tree - the Bridged Protocol Data Unit structure [21]. The structure of MST Configuration Identifier is shown further in Figure 2.9.

The description of MSTP BPDU structure [17]:

- Protocol Identifier – identifies the spanning tree family of protocols, value 0x00.
- Protocol Version Identifier – for the MSTP the value of this field is 3.
- BPDU Type – the RSTP and the MSTP share the same value for this field, which is 0x02.
- CIST Flags – the encoding of MSTP CIST flags is the same as for the RSTP and may be seen in Figure 2.5. The Topology Change Acknowledgement flag is not used in the MSTP CIST.
- CIST Root Identifier
- CIST External Root Path Cost – the path cost from the MST region of a sender to the MST region of the CIST root.
- CIST Regional Root Identifier – ID of the regional root.
- CIST Port Identifier – the Port Identifier of the sender bridge port.
- The following fields are same as in the RSTP BPDU: Message Age, Max Age, Hello Time, Forward Delay and Version 1 Length.
- Version 3 Length – Specifies the length of the fields specific for the MSTP. It is used to verify that an MSTP BPDU was received.
- MST Configuration Identifier – this field consists of the Configuration Identifier Format Selector, the Configuration Name, the Revision Level and the Configuration Digest, more described in Section 2.4.1, and shown in Figure 2.9.
- CIST Internal Root Path Cost – the path cost from the sender bridge port to the CIST regional root.
- CIST Bridge Identifier – the identifier of the sender bridge.
- CIST Remaining Hops – the remaining hops are decreased by one each time a BPDU is forwarded by a bridge. When the remaining hops are at the count of zero, the BPDU is discarded.
- MSTI Configuration Messages (optional) – contains MSTI Configuration Messages (maximum 64 Messages), more specific encoding of this field is described further in this section.

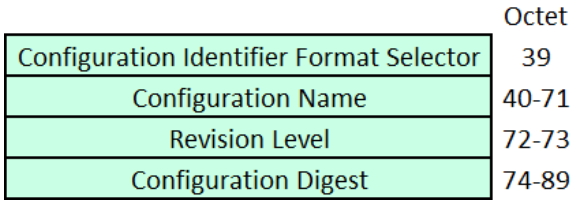

Figure 2.9: Multiple Spanning Tree BPDU – MST Configuration Identifier [21]

MSTI Configuration Messages

The structure of MSTI Configuration Messages is following, also displayed in Figure 2.10:

- MSTI Flags – the encoding of MSTI flags within the MSTI configuration message is displayed in Figure 2.11.
- MSTI Regional Root Identifier
- MSTI Internal Root Path Cost – the path cost from the local port to the MSTI regional root bridge.
- MSTI Bridge Identifier Priority – the priority of the designated bridge in the MSTI.
- MSTI Port Identifier Priority – the priority of the designated port in the MSTI.
- MSTI Remaining Hops – represents the value of remaining hops of the BPDU within the MSTI.

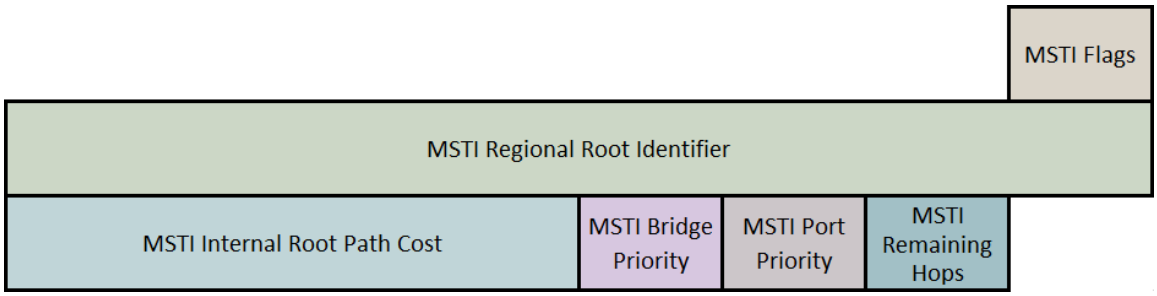

Figure 2.10: Multiple Spanning Tree BPDU – MSTI Configuration Messages [21].

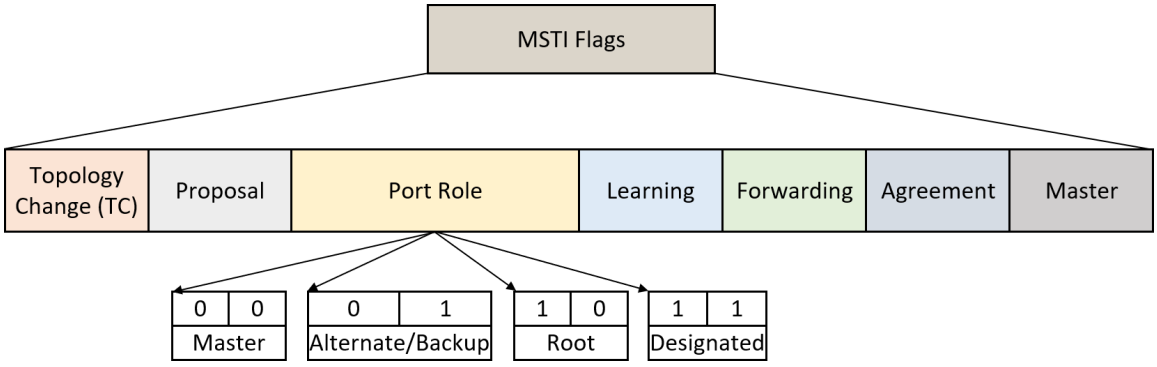

Figure 2.11: Multiple Spanning Tree BPDU – MSTI Configuration Message – MSTI Flags [21].

Chapter 3

Protocols Configuration on Cisco Devices

Configuration of STP protocols (RSTP, MSTP) is described in this chapter. Implemented solution will be tested in comparison with real functionality of these protocols configured on Cisco devices.

3.1 Monitoring and Maintaining STP protocols

While configuring the network, it is very important to have a possibility to monitor current state of the configuration within it. To check the configuration of the spanning tree protocols on a switched device, the following commands may be used [9, 2]:

- `Switch# show spanning-tree`
  - command displays STP parameters for all VLANs. Information about ports are summarized.
- `Switch# show spanning-tree detail`
  - command displays STP parameters for all VLANs. Information about ports are showed in detail.
- `Switch# show spanning-tree [vlan` *`vlan-id`*`] summary`
  - for each STP state, this command shows the total number of ports. which are configured in those states. When we want to show only the numbers of ports configured for each state within a VLAN, we may specify optional parameter `vlan` *`vlan-id`*.
- `Switch# show spanning-tree [vlan` *`vlan-id`*`] root`
  - displays the following information: the Bridge Identifier of the root, the root port, and the Root Path Cost.
- `Switch# show spanning-tree [vlan` *`vlan-id`*`] bridge`
  - information about the Bridge Identifier and STP timers for the local switch are shown with this command.

- `Switch# show spanning-tree interface` *`type port`*
  - shows information about the STP activity for a specific interface.
- `Switch# show running-config spanning-tree [all]`
  - shows the current configuration for spanning tree.
- `Switch# clear spanning-tree detected-protocol [interface` *`interface`* `[`*`interface-num | port-channel`*`]]`
  - the migration process for detected protocol (RSTP/MSTP) is restarted on the switch, or the specified interfaces of the switch. This forces the renegotiation with neighboring switches.

3.2 RSTP Configuration

In this section, we will look into the configuration settings for the RSTP on switched devices. The mentioned information are gained from the following source [3].

3.2.1 RSTP configuration commands

The RSTP protocol is now the default protocol running on switched devices. To explicitly configure the RSTP on a switched device, we need to enable the Rapid PVST+ mode on the device:

- `Switch(config)# spanning-tree mode rapid-pvst`

It is also possible to enable (and similarly disable) the Rapid PVST+ only for specified VLANs:

- `Switch(config)# spanning-tree` *`vlan-range`*
  - command for enabling Rapid PVST+ on the specified VLAN. The *vlan-range* parameter may be a value from the interval 2-4096 (except reserved VLAN values).
- `Switch(config)# no spanning-tree` *`vlan-range`*
  - command for disabling Rapid PVST+ on the specified VLAN.

After the RSTP is enabled on the switched device, we may also configure other parameters for the RSTP on the running Rapid PVST+ mode. To do so, the following commands may be used:

- `Switch(config)# spanning-tree vlan` *`vlan-range`* `root primary [diameter` *`dia`* `[hello-time` *`hello-time`*`]]`
  - with this command, the switch is configured as the primary root bridge. The default value for *dia* parameter is 7. The diameter specifies the maximum number of switches (hop counts) between any two end stations. The *hello-time* value can be configured from 1 to 10 seconds, default is 2 seconds.
- `Switch(config)# spanning-tree vlan` *`vlan-range`* `root secondary [diameter` *`dia`* `[hello-time` *`hello-time`*`]]`

  - with this command, the switch is configured as the secondary root bridge.

- `Switch(config-if)# spanning-tree [vlan` *`vlan-list`*`]`
  `port-priority` *`priority`*
  - through this command, the port priority for the LAN interface is configured. The default *priority* value is 128, and can be configured from one of the following values: 0, 32, 64, 128, 160, 192 and 224. Other values for this parameter are rejected. The lower value indicates the higher priority.

- `Switch(config)# spanning-tree pathcost method {long | short}`
  - configures the method used for path-cost calculations within the Rapid PVST+ mode. Default is short.

- `Switch(config-if)# spanning-tree [vlan` *`vlan-id`*`] cost [`*`value`*` | auto]`
  - with this command, the port cost for the LAN interface may be configured. For the short path-cost calculation method, the cost value may be configured from interval 1-65,535, and for the long method from 1-200,000,000. The default (auto) value is derived from the media speed of the interface.

- `Switch(config)# spanning-tree vlan` *`vlan-range`* `priority` *`value`*
  - the bridge priority of a VLAN is configured with this command. The priority *value* may be from range 0-61,440 in increments of 4096. Other values are rejected. The lower priority is considered better. The default value is 32,768.

- `Switch(config)# spanning-tree vlan` *`vlan-range`* `hello-time` *`hello-time`*
  - sets the hello time for a VLAN. Can be configured from range 1 to 10 seconds. The default value is 2 seconds.

- `Switch(config)# spanning-tree vlan` *`vlan-range`* `forward-time` *`forward-time`*
  - sets the forward delay time for a VLAN. Can be configured from range 4-30 seconds. The default value is 15 seconds.

- `Switch(config)# spanning-tree vlan` *`vlan-range`* `max-age` *`max-age`*
  - sets the maximum aging time for a VLAN. Can be configured from range 6-40 seconds. The default value is 20 seconds.

- `Switch(config-if)# spanning-tree link-type`
  `{auto | point-to-point | shared}`
  - type of a link is configured. The default is auto, which configures the link type according to the duplex mode of the interface. The half-duplex links are shared, the full-duplex links are point-to-point.

3.3 MSTP Configuration

In this section, we will discuss the possible methods for configuring the MSTP on a switch. Firstly, the MST needs to be enabled, then a various commands may be used for configuration of MST regions, its instances and other parameters.

3.3.1 MSTP Configuration Commands

A configuration of MSTP on a Cisco switch may be provided by using the commands specified in this section [2, 3].

Basic MSTP Configuration Commands

- `Switch(config)# spanning-tree mode mst`
  - enables MST on the switch.
- `Switch(config)# no spanning-tree mode mst`
  - disables MST on the switch and reset to the default Spanning Tree Protocol version.

MST Configuration Mode

The MST configuration mode serves to configure MCID (MST Configuration Identifier). For two bridges to belong to the same region they both need to have the same MCID, which consists of:

- MST configuration name,
- MST configuration revision number,
- instance to VLAN mapping.

The following configuration commands are used for the MST configuration:

- `Switch(config)# spanning-tree mst configuration`
  - enters configuration mode for the MST.
- `Switch(config-mst)# name` *`name`*
  - this command specifies the name for the MST configuration. The maximum length of *name* parameter is 32 characters, case sensitive.
- `Switch(config-mst)# revision` *`version`*
  - specifies the revision number for the MST configuration. The parameter *version* may be configured from interval 1–65,535.
- `Switch(config-mst)# instance` *`instance-id`* `vlan` *`vlan-range`*
  - maps VLAN(s) to an MST instance. The *instance-id* parameter represents the MSTID (Multiple Spanning Tree Instance Identifier), and may be configured from a range 0–4094 (where the instance number 0 represents configuration for the CIST, while the range 1–4094 represents other instances within specific region, the maximum number of configured instances within one region is 64, excluding the instance 0 for the CIST), the *vlan-range* may be configured from range 1–4094. The *vlan-range* parameter may be specified using hyphen as a range, or by using commas as a list of VLANs.

MSTP Bridge Parameters Configuration

The following commands are used to configure bridge times:

- `Switch(config)# spanning-tree mst hello-time` *`seconds`*
  - sets the hello time for all MST instances. Can be configured from range 1 to 10 seconds. The default value is 2 seconds.
- `Switch(config)# spanning-tree mst forward-time` *`seconds`*
  - sets the forward delay time for all MST instances. Can be configured from range 4 to 30 seconds. The default value is 15 seconds.
- `Switch(config)# spanning-tree mst max-age` *`seconds`*
  - sets the maximum-aging time for all MST instances. Can be configured from range 6 to 40 seconds. The default value is 20 seconds.
- `Switch(config)# spanning-tree mst max-hops` *`hop-count`*
  - specifies the number of hops in a region before discarding the BPDU. The *hop-count* may be from range 1–255. The default value is 20.
- `Switch(config)# spanning-tree transmit hold-count` *`hold-count`*
  - specifies the maximum transmission limit (maximum number of transmitted BPDUs within the hello time period) [4]. The *hold-count* may be from range 1–20. The default value is 6.

The bridge priority may be configured only separately for each tree instance. To do so, the following configuration commands may be used:

- `Switch(config)# spanning-tree mst` *`instance-id`* `priority` *`priority`*
  - configures the priority of the switch. The *priority* may be from range 0–61,440 in increments of 4096. Other values are rejected. The lower priority is considered better. The default value is 32,768.
- `Switch(config)# spanning-tree mst` *`instance-id`* `root primary [diameter` *`net-diameter`* `[hello-time` *`seconds`*`]]`
  - with this command a switch is configured as the root bridge. The *instance-id* parameter may be configured as a vlaue from range 0–4094. It may be specified as a single value, by using hyphen as a range, or by using commas as a list of instances. It is possible to specify the maximum number of switches between any two end stations by using the optional diameter, but only for the MST instance 0 (the CIST). The *net-diameter* parameter may be from range 2–7. The hello-time parameter may be configured from range 1–10, the default value is 2 seconds.
- `Switch(config)# spanning-tree mst` *`instance-id`* `root secondary [diameter` *`net-diameter`* `[hello-time` *`seconds`*`]]`
  - with this command a switch is configured as the secondary root bridge.

MSTP Port Parameters Configuration

The following commands may be used for configuration of the port priority and cost parameters for all tree instances on a specific interface:

- `Switch(config-if)# spanning-tree port-priority` *priority*
  - configures the port priority for all instances on this interface. The *priority* may be from range 0–240 in increments of 16. Other values are rejected. The lower priority is considered better. The default value is 128.
- `Switch(config-if)# spanning-tree cost [`*cost* `| auto]`
  - configures the cost on this interface. The *cost* may be from range 1–200,000,000. The default (auto) value is derived from the media speed of the interface. The auto value is not available on all types of Cisco devices. Another possibility to configure back to the default automatic cost calculation based on a media speed is to use the following command: `no spanning-tree cost`.

The port priority and cost parameters may be also configured individually for each tree instance. This configuration is preferred over the blanket configuration for all instances. The following commands are used for that matter:

- `Switch(config-if)# spanning-tree mst` *instance-id* `port-priority` *priority*
  - configures the port priority on this interface for the specified instance.
- `Switch(config-if)# spanning-tree mst` *instance-id* `cost [`*cost* `| auto]`
  - configures the cost on this interface for the specified instance. When the auto value is not available on a Cisco device, the following command may be used to configure back to the default automatic cost calculation based on a media speed: `no spanning-tree mst` *instance-id* `cost`.

The link type is configured in the same way as in the RSTP configuration 3.2.1.

3.3.2 Monitoring MSTP

To show the configuration for MST running on a switched device, and to monitor its activity within the network, we may use the following commands [2]:

- `Switch# show spanning-tree mst configuration`
  - shows configuration for the MST region.
- `Switch# show spanning-tree mst configuration digest`
  - shows the MD5 digest included in the current MST configuration identifier.
- `Switch# show spanning-tree mst` *instance-id*
  - shows MST configuration for the instance specified in this command.
- `Switch# show spanning-tree mst interface` *interface-id*
  - shows MST configuration for the interface specified in this command.

Chapter 4

Design and Implementation

We will focus on familiarization with technologies used for modeling and simulation of network processes, especially with OMNeT++ and INET, ANSAINET. Also a summary of existing solutions for STP implementations is contained in this chapter.

4.1 Technologies Used

4.1.1 OMNeT++

OMNeT++[1] (Objective Modular Network Testbed in C++) is an extensible, modular, component-based C++ simulation library and framework used primarily for building network simulators. It also includes integrated development environment which is based on the Eclipse[2] simulation IDE and extends it with additional functionality – possibility of creating and configuring models (by using NED and ini files), analyzing simulation results, etc. [1].

For network simulation, additional simulation models and frameworks must be used within OMNeT++ framework, because OMNeT++ itself does not contain any components specifically for computer networks [31]. The network simulation frameworks are implemented independently of the simulation framework OMNeT++. In this thesis, we will consider the following network simulation frameworks for OMNeT++ (from many others): the most common INET Framework 4.1.2, and an extension of the INET framework – ANSAINET 4.1.3.

OMNeT++ provides a generic component based architecture for representing network elements in the model. These model components are referred as modules. The communication between modules is based on message passing. The communication may be direct between two model components or may be transmitted via predefined connections. Depending on the model domain, message may refer to events, packets, commands, jobs, etc. [31]. To configure the simulation components in the OMNeT++ framework, the NED language and ini files are used.

The structure specification for a simulation model, declaration of simple modules and their interconnection with channels into compound modules may be ensured via the NED (Network Description) language. The configuration may be performed within a NED file. The NED language may be represented by an equivalent tree, thus it can be converted to

[1] OMNeT++, Discrete Event Simulator – https://omnetpp.org/
[2] Eclipse IDE – https://www.eclipse.org/

XML[3]. Parameters for the configuration, for example specifying the network, may be set within the ini file (the omnetpp.ini file by default).

4.1.2 INET

The INET framework[4] is an open-source network simulation framework developed for OMNeT++. It contains models for several protocols:

- for the link layer – Ethernet, PPP, IEEE 802.11, IEEE 802.1d, etc.,
- for the network layer – IPv4, IPv6, BGP, OSPF, RIP, etc.,
- for the transport layer – TCP, UDP, SCTP, etc.

4.1.3 ANSAINET

The Automated Network Simulation and Analysis (ANSA)[5] is a project developed at Faculty of Information Technology at the Brno University of Technology. The source code of this project is referred as ANSAINET, because this project is an extension to the INET framework for the network simulation.

4.2 State of the Art

In this section, we will look into current state of implementation of the STP family protocols for INET framework. The version of INET framework observed is INET 4.2.0[6].

The implementations of the basic STP and the RSTP are already contained within the INET framework. The Multiple Spanning Tree Protocol implementation is not implemented yet. We will look into the RSTP implementation in more detail further in this section.

4.2.1 RSTP Implementation

The current implementation of RSTP protocol (IEEE 802.1D-2004) is contained within the INET package `inet.linklayer.ieee8021d.rstp`. According to the statement in the implementation of the RSTP module[7], the implementation of RSTP is complete except it does not fall back to STP, when the RSTP is not supported by peers. The following source files are contained within the RSTP package:

- `Rstp.h/cc` – this class contains implementation of the RSTP functionality. It contains methods for initializing ports, sending and processing BPDUs, comparing BPDUs, comparing RSTP data, printing data info, handling timers for RSTP, checking topology changes and other associated methods.
- `Rstp.ned` – contains the description of the RSTP protocol, its interfaces and parameters. Also includes the location of the package within the INET framework. It reconnects the source code of the protocol with the simulated Ethernet switch devices.

[3] Varga, A. OMNeT++ Simulation Manual – https://doc.omnetpp.org/omnetpp/manual/
[4] INET Framework – https://inet.omnetpp.org/
[5] ANSA – https://ansa.omnetpp.org/
[6] Available at https://github.com/inet-framework/inet/archive/v4.2.0.zip.
[7] RSTP module implementation – https://github.com/inet-framework/inet/blob/master/src/inet/linklayer/ieee8021d/rstp/Rstp.ned

The RSTP protocol also uses source files from the `inet.linklayer.ieee8021d.common` package, which includes common implementation methods for both, STP protocol (IEEE 802.1D-1998) and the RSTP protocol. The package comprises following:

- `StpBase.h/cc` – this class stores the parameter values for timers, bridge address and priority, interfaces, MAC address table, etc.
- `Ieee8021dBpdu.msg` – specifies the Bridge Protocol Data Units for the STP and the RSTP.
- `Ieee8021dBpduSerializer.h/cc` – this class implements serialization methods for converting between BPDU messages and their binary format.

4.2.2 Integration of STP Protocols

The spanning tree protocols operate on the second layer of the OSI model, the data link layer. One of the main components for the data link layer is the Ethernet switch. For the OMNeT++ simulator, the Ethernet switch compound module is a part of the `inet.node.ethernet` package. The `EtherSwitch` compound module is displayed in Figure 4.1. The module labeled as stp on the mentioned figure represents the `ISpanningTree` module interface. Both, the STP and the RSTP implementations inherit from this module.

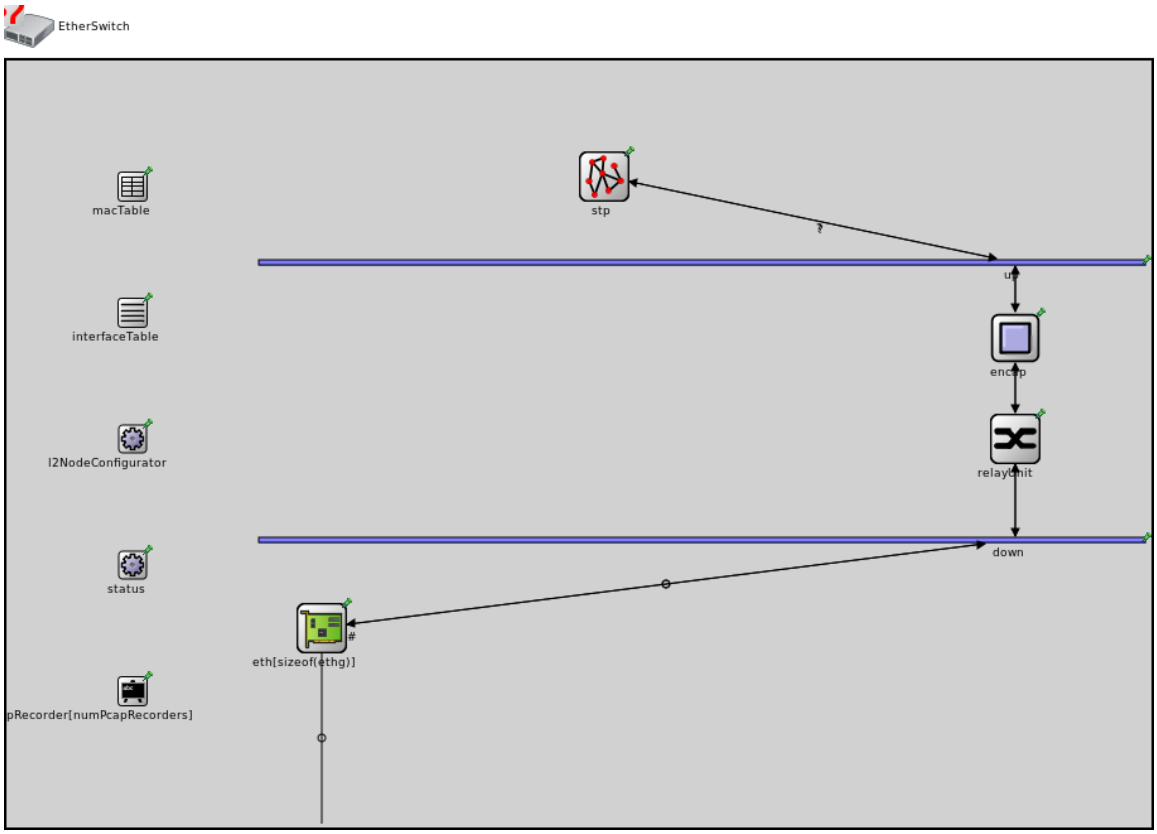

Figure 4.1: Ethernet switch compound module[8], taken over from [7].

The `ISpanningTree` module interface is specified within the `ISpanningTree.ned` file. It contains definition of common parameters for all modes of the Spanning Tree Protocol

(STP, RSTP). However, the module interface cannot contain the specification of the default values for those parameters. Therefore, the default parameter values, and parameters specific to a particular mode of the STP are specified within the corresponding protocol modules. For the RSTP it is the `Rstp.ned` file, and it contains specification for the following RSTP parameters: `helloTime`, `forwardDelay`, `maxAge`, `migrateTime`, `tcWhileTime`, `bridgePriority` and `autoEdge`.

The Spanning Tree Protocols make use of per-port configuration data. The state of a port is monitored within the `InterfaceTable` module. Parameters for the port (link cost, port priority and edge flag) are configured on the `L2NetworkConfigurator` module via the XML (Extensible Markup Language) configuration file. The protocol operation is also based on the usage of MAC addresses for the network modules configured in the simulated topology. The MAC addresses are managed by the `MacAddressTable` module. Path to both of these modules is also one of the configuration parameters for the Spanning Tree Protocols.

4.2.3 Configuration

To simulate a specific network, it is possible to use the `.ini` file (Initialization file). The default initialization file used for this purpose is the `omnetpp.ini` file. It is used to pass parameters to the model used in the simulated network.

The configuration of port data is done by using the XML configuration file, as already mentioned in 4.2.2. An example configuration for STP and RSTP modules is shown in Figure 4.2.

```
<config>
    <interface hosts='switch1' ports='1' cost='10' priority='240'/>
    <interface hosts='switch3' ports='0' cost='10' priority='32'/>
</config>
```

```
<config>
    <interface hosts='**' ports='**' cost='19' priority='128'/>
</config>
```

Figure 4.2: An example of XML interface configurations for STP and RSTP. The first configuration sets priority and cost parameters for specified switch and port, the second configuration sets these parameters on all switched devices and all their ports in the simulated network.

4.3 Implementation Notes

The main goal is to implement the MSTP into the INET framework. The design is based on the IEEE Standard 802.1Q, version 2018[12]. The standard describes also interoperability between the RSTP, MSTP and ISIS-SPB (Intermediate System to Intermediate System Protocol for Shortest Path Bridging). The implemented functionality will exclude the operations for ISIS-SPB and interoperability between protocols will be also omitted. The

L2GP (Layer 2 Gateway Port) functionality, also described in the mentioned standard is not implemented. The implemented solution is placed on GitHub[9].

Since the MSTP is related to other Spanning Tree Protocols, we will also use the common modules described in 4.2.2. The MSTP will be implemented into the `Mstp` module which will run under `EthernetSwitch`, the same compound module used by the STP or RSTP. It will also inherit from the `ISpanningTree` module interface.

The `L2NetworkConfigurator` module will be enhanced with the configuration of MSTP properties for ports. All of the MSTP parameters, which may be configured for a port have the same meaning for the MSTP as described in the IEEE Standard 802.1Q, version 2018. An example of a configuration for MSTP port parameters is showed in Figure 4.3.

```
<config>
    <interface hosts='**' ports='**' cost='2121212' priority='4096'/>

    <interface hosts='switch1' ports='1'>
        <instance id='300' priority='64'/>
    </interface>
    <interface hosts='switch1' ports='3' cost='500000'/>
</config>
```

Figure 4.3: An example of XML interface configurations for MSTP. In the first step, configures the default port cost and priority for all ports on all switches. Then the default port priority configuration is replaced on port 1 of the switch 1 for instance with `MSTID` 300. Then the default cost on port 3 of the switch 1 is replaced for all instances by the value 500,000.

The `L2NetworkConfigurator` module is connected to the `L2NodeConfigurator` module, contained within the `EtherSwitch` module. The following parameters may be specified inside the XML configuration and porocessed by this module:

- `AdminEdge` – if set, the port is determined as operEdge port immediately on initialization, without a delay to detect other switched devices in the simulated topology,
- `AutoEdge` – if set, automatically determines the port as operEdge after recognizing other switched devices in the simulated topology,
- `AutoIsolate` – if set, conditionally causes a Designated Port to transit into Discarding state,
- `mcheck` – forces MST BPDU transitions for the MigrateTime period, to test if any STP bridge is connected to the switched device (since the interoperability between protocols will be omitted, this parameter will not affect the MSTP functionality),
- port path cost – the `cost` parameter for a port, configured either as a default value per port, or separately for each instance (this instance must be also configured on the `Mstp` module, configuration for the `Mstp` module described further in this section),

[9]GitHub repository – https://github.com/simona5108/inet/tree/ieee8021s

[9]Figure of the Ethernet switch, source – https://doc.omnetpp.org/inet/api-current/neddoc/inet.node.ethernet.EtherSwitch-type.svg

- port priority – the `priority` parameter for a port, configured either as a default value per port, or separately for each instance (same as for a cost parameter, the instance must be also configured on the `Mstp` module).

The `Mstp` module configuration includes specifying the following parameters: `forwardDelay`, `maxAge`, `txHoldCount`, `maxHops`, `bridgePriority`, `mstConfig` parameter, which represents an XML configuration for instance to VLAN mappings and `mstiBPConfig` for Bridge Priority configuration per specified MSTI. The `helloTime` parameter may not be changed for MSTP according to the IEEE Standard 802.1Q (2018). An example of MST XML configuration parameters is showed in Figure 4.4 for `mstConfig` and in Figure 4.5 for `mstiBPConfig` parameter.

```
<mstConfig name='REGION1234' revision='0'>
    <instance id='1' vlan='1'/>
    <instance id='2' vlan='2'/>
    <instance id='300' vlan='3-300'/>
</mstConfig>
```

Figure 4.4: Configuration for an MST Region applied to an `Mstp` module running on a switch. An MST Region with the Configuration Name `REGION1234` and the Revision Level 0 will be created. The Configuration Digest for this region will be computed based on the stated instance to VLAN mapping.

```
<mstiBPConfig>
    <instance id='300' priority='16384'/>
</mstiBPConfig>
```

Figure 4.5: Configuration of a bridge priority parameter on a switch per instance.

The implemented solution includes logic for mapping instances to VLANs. It is not directly connected to any VLAN module yet. One of the main functionality for MSTP is that the switches in simulated topology belong to one MST Region. For two switches to belong to the same region, they must have the same MST Configuration Identifier, which consists of an MST Configuration Identifier Format Selector (fixed value), the Configuration Name, the Revision Level and the Configuration Digest. The Configuration Digest is created from the MST Configuration Table, which consists of the instance to VLAN mapping. It is computed as a HMAC-MD5 (Hash-based Message Authentication Code, Message Digest Algorithm 5) signature. The implemented solution is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm[10] (RFC 1321) and the HMAC-MD5 method[11] (RFC 2104).

The operation of the MSTP is implemented through state machines based on the IEEE Standard 802.1Q (2018). These state machines include the following:

- Port Timers SM
- Port Receive SM

[10] RSA Data Security, Inc. MD5 Message-Digest Algorithm – https://tools.ietf.org/html/rfc1321
[11] HMAC-MD5 – https://tools.ietf.org/html/rfc2104

- Port Protocol Migration SM
- Bridge Detection SM
- Port Transmit SM
- Port Information SM
- Port Role Selection SM
- Port Role Transitions SM – includes state machines for port transitions to each port role: Disabled port role, Master port role, Root port role, Designated port role and one SM for both, the Alternate port role and the Backup port role. The connection of all of the port role transition state machines into one SM is not specified in the standard. The implemented solution includes the generalized SM which comprises all of the stated port role transition SM. The SM is depicted in Appendix B.
- Port State Transition SM
- Topology Change SM

The interconnection of these state machines is showed in Figure 4.6. As already mentioned, the L2 Gateway Port functionality (therefore also the L2 Gateway Port Receive SM) is excluded from the implementation as well as the original Figure.

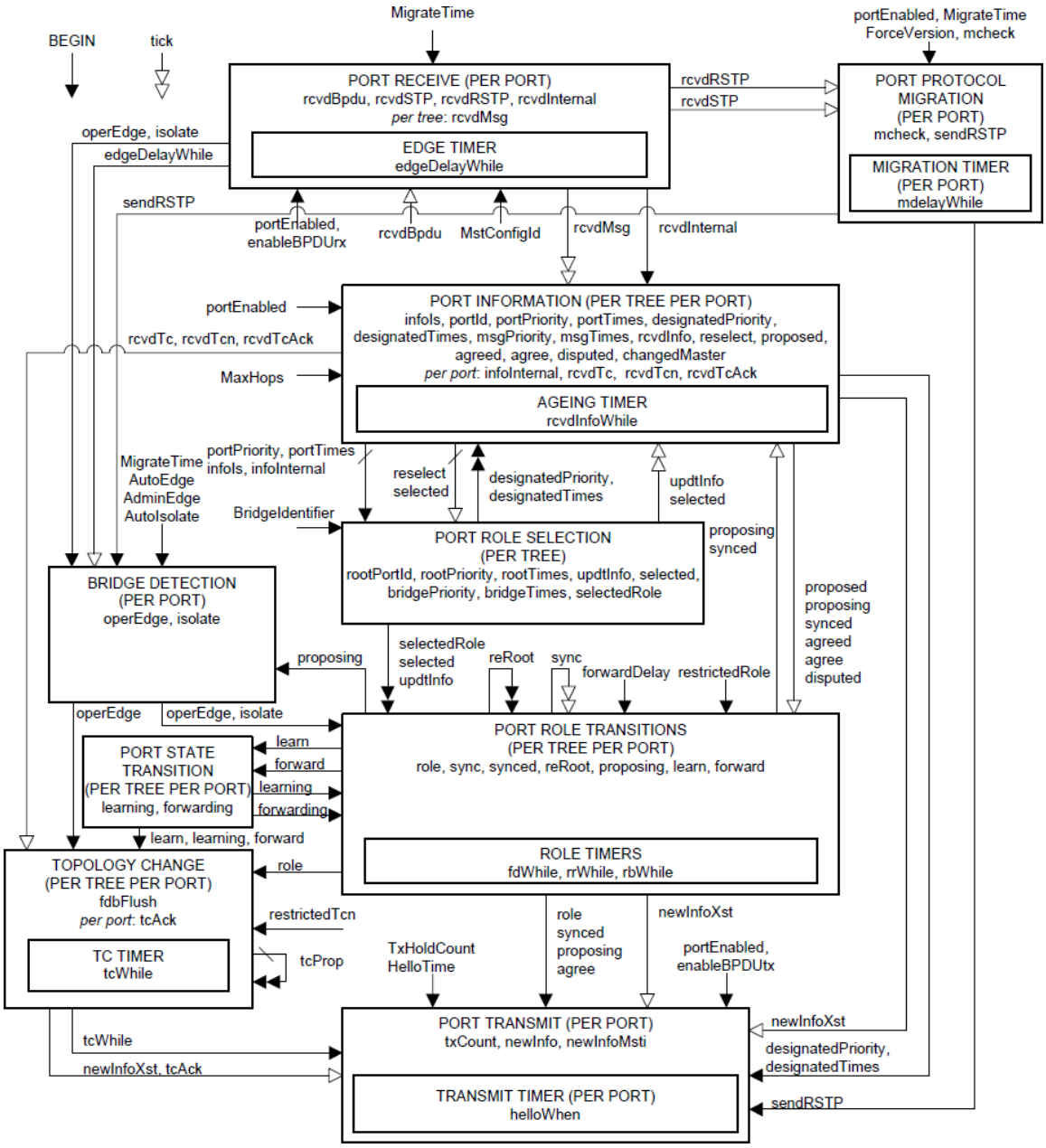

Figure 4.6: Spanning tree protocol state machines—overview and relationships, adopted from [12, p. 508, Figure 13-13]

Chapter 5

Testing

This chapter concerns with testing of the MSTP protocol implemented as the `Mstp` module into the INET framework. Simulated solution in the OMNeT++ framework will be compared with the referential behavior of the Cisco switched devices. The same simulated topology will be used for both cases. The referential behavior will be observed in EVE-NG (The Emulated Virtual Environment – Next Generation) testing environment.

This chapter is divided into three main sections. In the first section there is introduced the topology used for the testing in detail. It is followed by the section about the tree structure establishment for the MSTP. And the last part describes the link failure and reestablishment scenario and behavior of the MSTP afterwards.

5.1 Testing Topology

The topology used for testing the solution is displayed in Figure 5.1. It consists of eight switched devices with MSTP configured. The switched devices labeled as `S1-S4` belongs to the MST Region `REGION1234`, switches labeled as `S5-S7` to `REGION567` and the switch `S8` is in a region of its own (not configured, only default region parameters).

To show the correctness of the implementation better, the following configuration steps will be performed:

- The costs on links between `S1` and `S6` and between `S5` and `S6` will be set to a quarter of their default values, to ensure the path between regions will lead just through the link between `S1` and `S6`. Therefore, the switch `S6`. The cost value for a link is determined based on a link speed by default. (For a link with a link speed of 10 Mb/s the default cost value is 2,000,000.) To also validate the cost configuration per instance, the instance `MST200` in MST Region `REGION567` will be modified on links `S5` to `S6` and `S5` to `S7` by setting the cost to a smaller value (etc. 1000), so the preferred path from `S6` to `S7` will go through `S5`.

- To validate the configuration parameter for bridge priority, there will be different instances configured in both MST Regions so that each switch will be set as root in one of the instances. The instances for both regions are displayed in Figure 5.2 for the MST Region `REGION1234` and in Figure 5.3 for the MST Region `REGION567`. Also the expected tree structure determination is depicted in the mentioned figures.

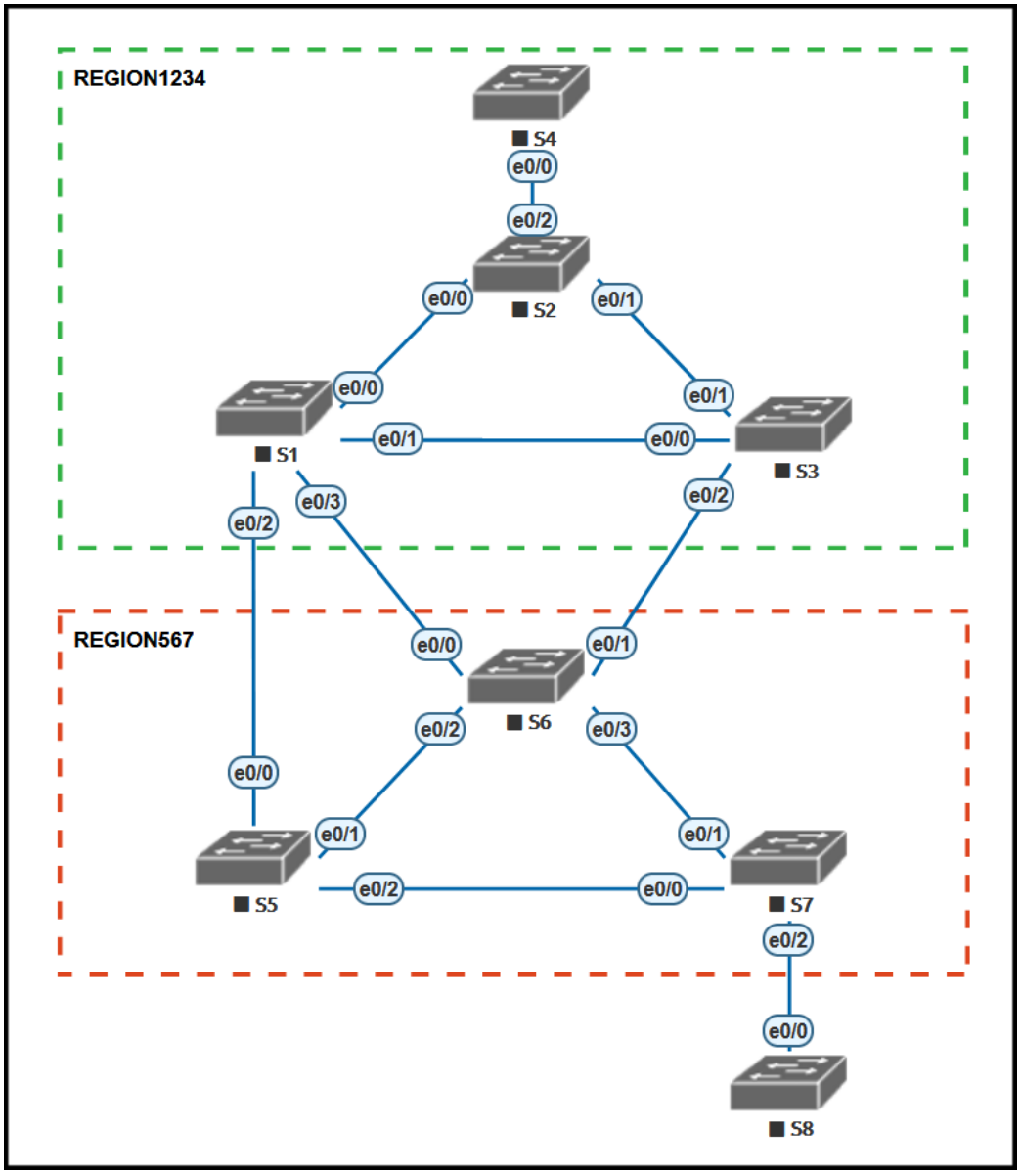

Figure 5.1: Testing Topology with depicted MST Regions

- To verify the functionality of port priority parameter (configured in XML configuration for the `L2NetworkConfigurator` module), the instance created for the region `REGION1234`, labeled as instance `MST300` will be modified for links between `S1`, `S3` and `S3`, `S2`. That will not affect the topology determination, but we may notice the changed value while monitoring the MSTP and in transmitted BPDUs for that MST instance.
- The integration of switched devices into the mentioned regions depends also on the instance to VLAN mappings. The implementation provides the possibility to configure the mapping between the MST instances and VLANs, even though the implemented `Mstp` module is not yet connected to VLAN module. Also the computation of the MST Configuration Digest is implemented, and its correctness will be verified based on the configured mapping.

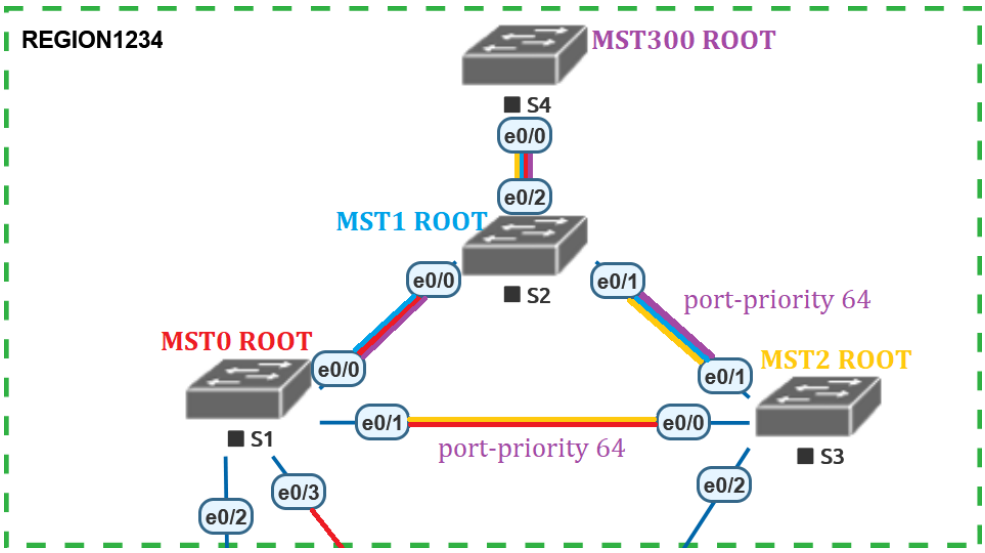

Figure 5.2: Tree structure inside region `REGION1234` after topology establishment. All switches stated as `MSTx ROOT` are the MSTI Regional Roots for instance `x`. The switch `S1` is also the CIST Root for the testing topology.

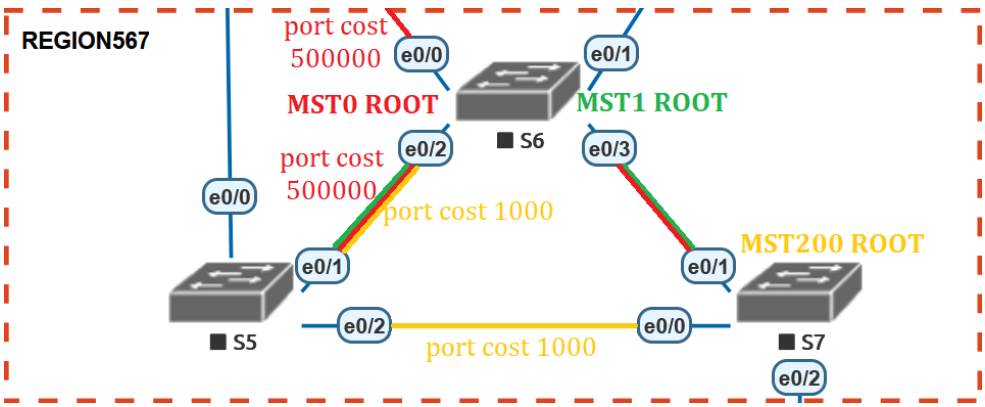

Figure 5.3: Tree structure inside region `REGION567` after topology establishment. All switches stated as `MSTx ROOT` are the MSTI Regional Roots for instance `x`.

5.2 Tree Structure Establishment

In this section, we will look into the establishment of the topology described in Section 5.1. A comparison of BPDU format captured from a Cisco switch in contrast to a packet received in simulated `Mstp` module will be made.

5.2.1 BPDU Format Comparison

After the switched devices are started, they begin to transmit the BPDU packets to establish a loop-free topology. An example of the content of the transmitted BPDU for the simulated solution in comparison with captured packet on a Cisco switch is displayed in Figure 5.4 and 5.5 for the main MSTP BPDU content (excluded MSTI messages) and in Figure 5.6 and 5.7 for the MSTI messages. All figures display a packet received on the switch `S2` from the switch `S3` in the testing topology.

By comparing the MSTI content of the received packets in Figure 5.6 and 5.7, we may also verify the functionality of the MST Configuration digest computation. We can also make sure, that the costs for individual MSTI messages is properly calculated.

frame	const inet::Ptr<inet::MSTBpdu const> &	{...}
p	const inet::MSTBpdu *	0x1e2d8200
inet::BpduBase	inet::BpduBase	{...}
tcaFlag	bool	false
agreementFlag	bool	false
forwardingFlag	bool	false
learningFlag	bool	false
portRole	inet::PortRole	inet::ALTER_OR_BACKUP
proposalFlag	bool	false
tcFlag	bool	false
cistRootPriority	uint16_t	32768
cistRootAddress	inet::MacAddress	{...}
address	uint64_t S1: 0A-AA-00-00-00-01	11725260718081
cistExternalPathCost	uint32_t	0
cistRegionalRootPriority	uint16_t	32768
cistRegionalRootAddress	inet::MacAddress	{...}
address	uint64_t S1: 0A-AA-00-00-00-01	11725260718081
portPriority	uint8_t	128 '\200'
portNum	uint8_t	101 'e'
messageAge	omnetpp::simtime_t	{...}
maxAge	omnetpp::simtime_t	{...}
helloTime	omnetpp::simtime_t	{...}
forwardDelay	omnetpp::simtime_t	{...}
versionOneLen	uint8_t	0 '\0'
versionThreeLen	uint16_t	112
formatSelector	uint8_t	0 '\0'
configurationName	omnetpp::opp_string	{...}
revisionLevel	uint16_t	0
configurationDigest	omnetpp::opp_string	{...}
cistInternalPathCost	uint32_t	2000000
bridgePriority	uint16_t	32768
bridgeAddress	inet::MacAddress	{...}
address	uint64_t S3: 0A-AA-00-00-00-08	11725260718088
remainingHops	uint8_t	19 '\023'
mstiMsgs	inet::MSTIMsgsVec	{...}
[0]	inet::MSTIMsg	{...}
[1]	inet::MSTIMsg	{...}
[2]	inet::MSTIMsg	{...}

Figure 5.4: The format of a transmitted BPDU (part 1) in the implemented solution in OMNeT++.

```
> Frame 17: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0
> IEEE 802.3 Ethernet
> Logical-Link Control
v Spanning Tree Protocol
    Protocol Identifier: Spanning Tree Protocol (0x0000)
    Protocol Version Identifier: Multiple Spanning Tree (3)
    BPDU Type: Rapid/Multiple Spanning Tree (0x02)
  v BPDU flags: 0x04, Port Role: Alternate or Backup
      0... .... = Topology Change Acknowledgment: No
      .0.. .... = Agreement: No
      ..0. .... = Forwarding: No
      ...0 .... = Learning: No
      .... 01.. = Port Role: Alternate or Backup (1)
      .... ..0. = Proposal: No
      .... ...0 = Topology Change: No
  > Root Identifier: 32768 / 0 / aa:bb:cc:00:10:00
    Root Path Cost: 0
  > Bridge Identifier: 32768 / 0 / aa:bb:cc:00:10:00
    Port identifier: 0x8002
    Message Age: 0
    Max Age: 20
    Hello Time: 2
    Forward Delay: 15
    Version 1 Length: 0
    Version 3 Length: 112
  v MST Extension
      MST Config ID format selector: 0
      MST Config name: REGION1234
      MST Config revision: 0
      MST Config digest: e04722e477d04ac81bcc97fbad89c169
      CIST Internal Root Path Cost: 2000000
    > CIST Bridge Identifier: 32768 / 0 / aa:bb:cc:00:30:00
      CIST Remaining hops: 19
    > MSTID 1, Regional Root Identifier 16384 / aa:bb:cc:00:20:00
    > MSTID 2, Regional Root Identifier 16384 / aa:bb:cc:00:30:00
    > MSTID 300, Regional Root Identifier 16384 / aa:bb:cc:00:40:00
```

Figure 5.5: The format of a transmitted BPDU (part 1) on a Cisco switch captured in Wireshark.

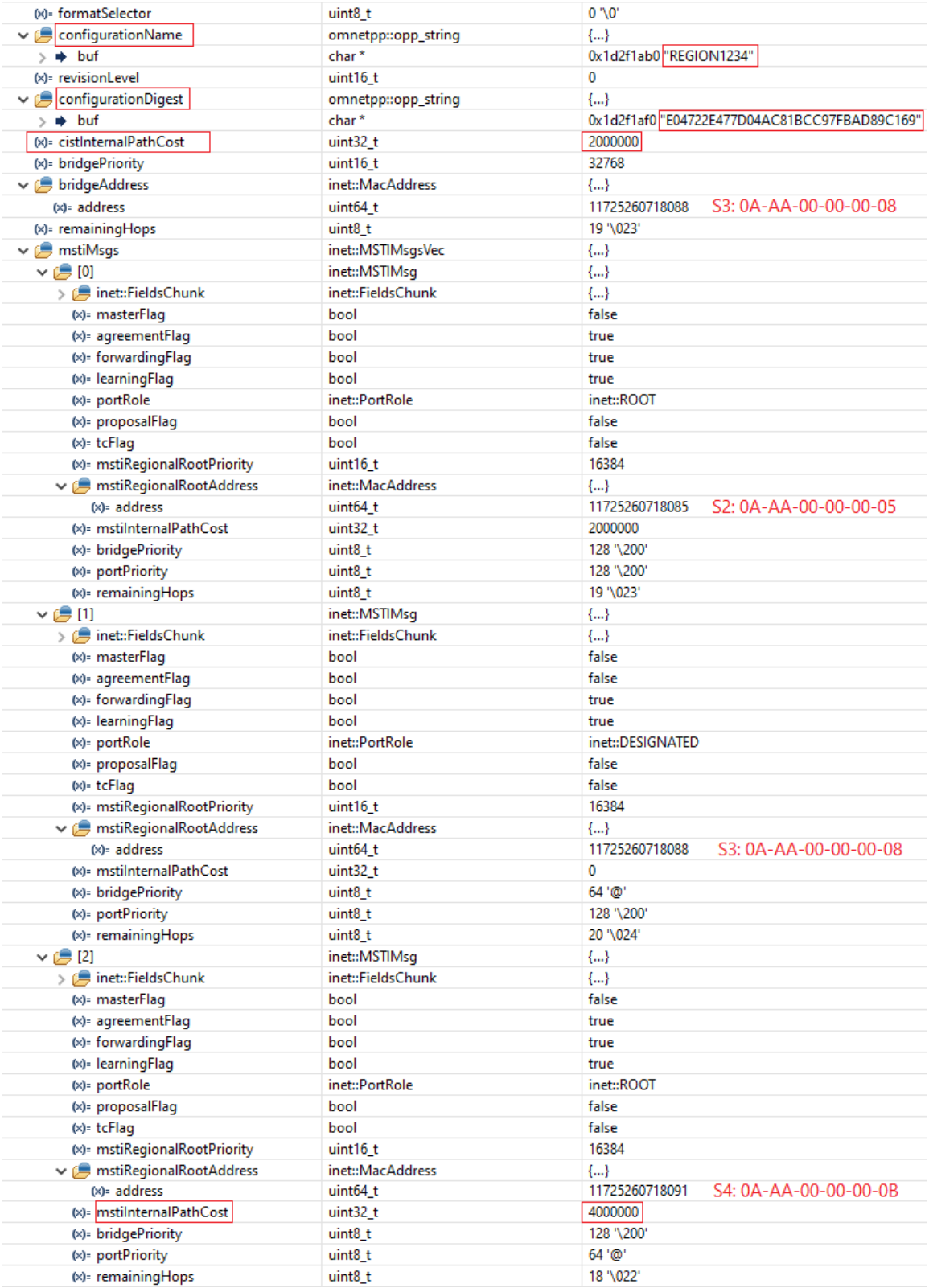

Figure 5.6: The format of a transmitted BPDU (part 2) in the implemented solution in OMNeT++. We may see the details for the transmitted MSTI messages.

```
MST Extension
    MST Config ID format selector: 0
    MST Config name: REGION1234
    MST Config revision: 0
    MST Config digest: e04722e477d04ac81bcc97fbad89c169
    CIST Internal Root Path Cost: 2000000
  > CIST Bridge Identifier: 32768 / 0 / aa:bb:cc:00:30:00
    CIST Remaining hops: 19
  v MSTID 1, Regional Root Identifier 16384 / aa:bb:cc:00:20:00
    > MSTI flags: 0x38, Forwarding, Learning, Port Role: Root
      0100 .... = Priority: 0x4
      .... 0000 0000 0001 = MSTID: 1
      Regional Root: aa:bb:cc:00:20:00 (aa:bb:cc:00:20:00)
      Internal root path cost: 2000000
      Bridge Identifier Priority: 8
      Port identifier priority: 8
      Remaining hops: 20
  v MSTID 2, Regional Root Identifier 16384 / aa:bb:cc:00:30:00
    v MSTI flags: 0x7c, Agreement, Forwarding, Learning, Port Role: Designated
        0... .... = Topology Change Acknowledgment: No
        .1.. .... = Agreement: Yes
        ..1. .... = Forwarding: Yes
        ...1 .... = Learning: Yes
        .... 11.. = Port Role: Designated (3)
        .... ..0. = Proposal: No
        .... ...0 = Topology Change: No
      0100 .... = Priority: 0x4
      .... 0000 0000 0010 = MSTID: 2
      Regional Root: aa:bb:cc:00:30:00 (aa:bb:cc:00:30:00)
      Internal root path cost: 0
      Bridge Identifier Priority: 4
      Port identifier priority: 8
      Remaining hops: 20
  v MSTID 300, Regional Root Identifier 16384 / aa:bb:cc:00:40:00
    v MSTI flags: 0x38, Forwarding, Learning, Port Role: Root
        0... .... = Topology Change Acknowledgment: No
        .0.. .... = Agreement: No
        ..1. .... = Forwarding: Yes
        ...1 .... = Learning: Yes
        .... 10.. = Port Role: Root (2)
        .... ..0. = Proposal: No
        .... ...0 = Topology Change: No
      0100 .... = Priority: 0x4
      .... 0001 0010 1100 = MSTID: 300
      Regional Root: aa:bb:cc:00:40:00 (aa:bb:cc:00:40:00)
      Internal root path cost: 4000000
      Bridge Identifier Priority: 8
      Port identifier priority: 4
      Remaining hops: 19
```

Figure 5.7: The format of a transmitted BPDU (part 2) on a Cisco switch captured in Wireshark. We may see the details for the transmitted MSTI messages.

5.2.2 Monitoring MSTP

To monitor the progress of the protocol during the simulation run, there is possibility to use the `WATCH` macros provided by OMNeT++. By default, it includes all primitive types and the `std::string` type, but it may be also used with any object, which contains an implementation for the stream output operator.

For the MSTP protocol, the most significant information to be observed are available from the `MstpPortData` class which contains data for individual interfaces. In Figure 5.8 and 5.9, there is shown an example of the monitoring possibilities while running the simulation. The first figure shows the initialization of the data after starting the simulation, while in the second figure we may see how this data were changed for the same port after topology establishment. To make a comparison with a referential behavior, there are displayed MSTP data in Cisco switch in Figure 5.10 using the Cisco command `show spanning-tree mst`. The `MstpPortData` are contained within the `L2NodeConfigurator` module.

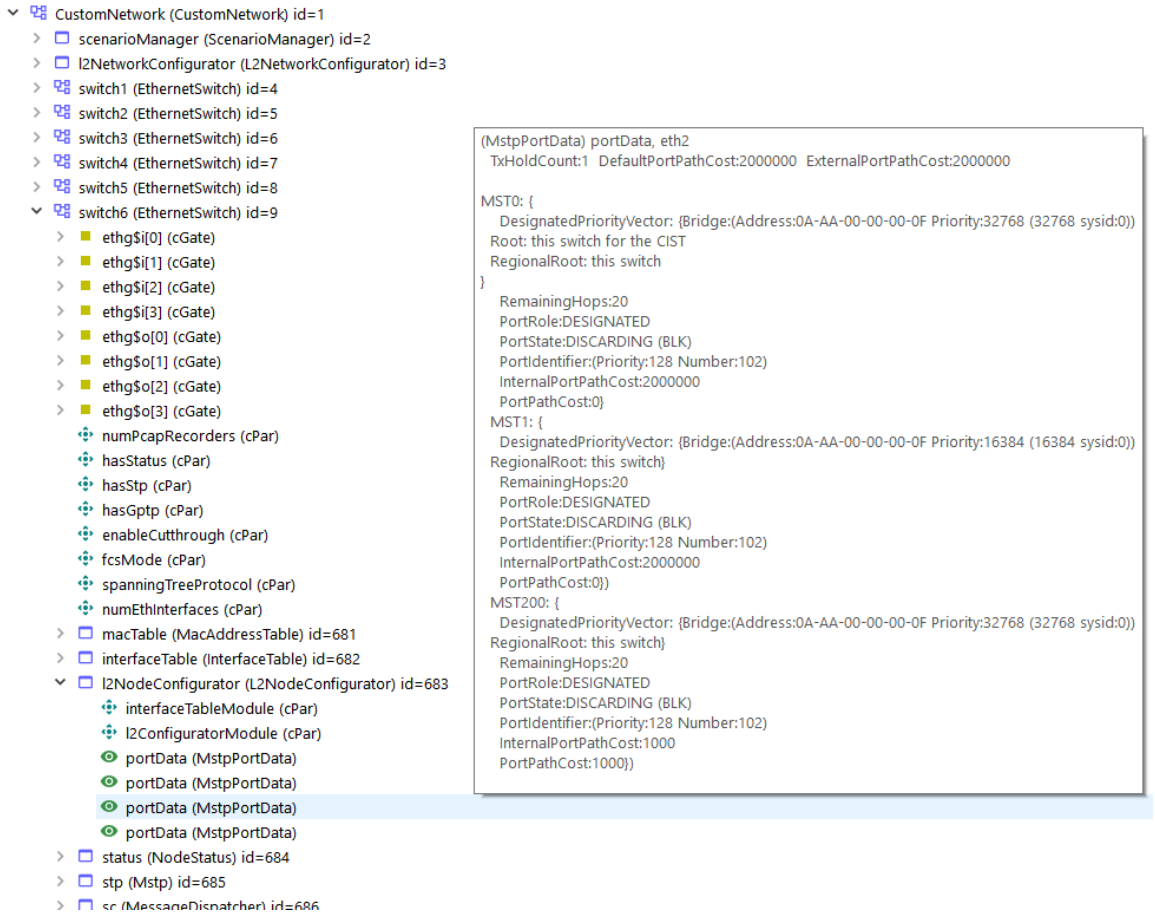

Figure 5.8: MSTP port data for the Ethernet interface (`eth2`) on switch `S6` available after topology initialization while running the simulation example. The `eth2` interface leads towards the switch `S5`. We may notice that the configuration was loaded, for example by looking at the instance `MST1` and the `Priority` parameter is configured to the value 16384. We may also see changed port path cost for the instance `MST200`.

```
(MstpPortData) portData, eth2
  TxHoldCount:5  DefaultPortPathCost:2000000  ExternalPortPathCost:2000000

MST0: {
    DesignatedPriorityVector: {Bridge:(Address:0A-AA-00-00-00-0F Priority:32768 (32768 sysid:0))
  Root: (Address:0A-AA-00-00-00-01 Priority:32768 (32768 sysid:0))  ExternalRootPathCost: 2000000
  RegionalRoot:(Address:0A-AA-00-00-00-0C Priority:32768 (32768 sysid:0))  InternalRootPathCost:2000000
}
    RemainingHops:19
    PortRole:ROOT
    PortState:FORWARDING
    PortIdentifier:(Priority:128 Number:102)
    InternalPortPathCost:2000000
    PortPathCost:0}
  MST1: {
    DesignatedPriorityVector: {Bridge:(Address:0A-AA-00-00-00-0F Priority:16384 (16384 sysid:0))
  RegionalRoot: this switch}
    RemainingHops:20
    PortRole:DESIGNATED
    PortState:FORWARDING
    PortIdentifier:(Priority:128 Number:102)
    InternalPortPathCost:2000000
    PortPathCost:0})
  MST200: {
    DesignatedPriorityVector: {Bridge:(Address:0A-AA-00-00-00-0F Priority:32768 (32768 sysid:0))
  RegionalRoot:(Address:0A-AA-00-00-00-13 Priority:16384 (16384 sysid:0))  InternalRootPathCost:2000}
    RemainingHops:18
    PortRole:ROOT
    PortState:FORWARDING
    PortIdentifier:(Priority:128 Number:102)
    InternalPortPathCost:1000
    PortPathCost:1000})
```

Figure 5.9: MSTP port data for the Ethernet interface (`eth2`) on switch `S6` available after topology establishment while running the simulation example. The `eth2` interface leads towards the switch `S5`. Compared to the information showed in Figure 5.8 we may notice how the `Root` address has changed to the address of the switch `S1` in the CIST instance `MST0`. There is also changed the `RegionalRoot` in instance `MST200`, because of the `InternalRootPathCost` 2000. Also the port roles and port states are established.

```
S6#sh spanning-tree mst

##### MST0    vlans mapped:   201-4094
Bridge        address aabb.cc00.6000  priority      32768 (32768 sysid 0)
Root          address aabb.cc00.1000  priority      32768 (32768 sysid 0)
              port    Et0/0           path cost     500000
Regional Root this switch
Operational   hello time 2 , forward delay 15, max age 20, txholdcount 6
Configured    hello time 2 , forward delay 15, max age 20, max hops    20

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Et0/0            Root FWD 500000    128.1    Shr Bound(RSTP)
Et0/1            Altn BLK 2000000   128.2    Shr Bound(RSTP)
Et0/2            Desg FWD 2000000   128.3    Shr
Et0/3            Desg FWD 2000000   128.4    Shr

##### MST1    vlans mapped:   1
Bridge        address aabb.cc00.6000  priority      16385 (16384 sysid 1)
Root          this switch for MST1

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Et0/0            Mstr FWD 500000    128.1    Shr Bound(RSTP)
Et0/1            Altn BLK 2000000   128.2    Shr Bound(RSTP)
Et0/2            Desg FWD 2000000   128.3    Shr
Et0/3            Desg FWD 2000000   128.4    Shr

##### MST200  vlans mapped:   2-200
Bridge        address aabb.cc00.6000  priority      32968 (32768 sysid 200)
Root          address aabb.cc00.7000  priority      16584 (16384 sysid 200)
              port    Et0/2           cost          2000      rem hops 18

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Et0/0            Mstr FWD 500000    128.1    Shr Bound(RSTP)
Et0/1            Altn BLK 2000000   128.2    Shr Bound(RSTP)
Et0/2            Root FWD 1000      128.3    Shr
Et0/3            Altn BLK 2000000   128.4    Shr
```

Figure 5.10: MSTP port data for the Ethernet interface (`eth2`) on switch `S6` available after topology establishment displayed in the Cisco switch.

To see the correctness of the configuration for multiple spanning tree instances and their mapping to VLANs, there is an `MstConfigId` watchable object within the `Mstp` module. For region `REGION567` on the switch `S6` the MST Configuration Identifier is displayed in Figure 5.11. We may also compare it with the referential configuration on the Cisco switch `S6` shown in Figure 5.10. In the MST Configuration Identifier we are looking at the following parameters: Configuration Name, Revision Level and Configuration Digest. To compute the Configuration Digest, the MD5 Message-Digest Algorithm[1]. The purpose of observing

[1]RSA Data Security, Inc. MD5 Message-Digest Algorithm – https://tools.ietf.org/html/rfc1321

the MST Configuration Identifier is to determine that two switches belong to the same MST Region.

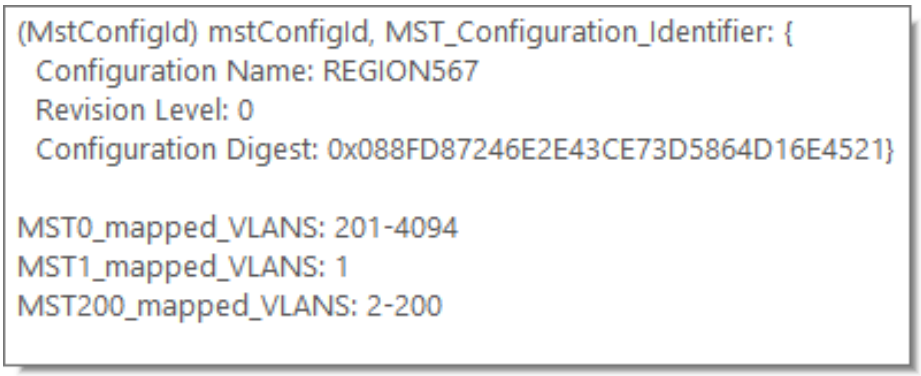

Figure 5.11: The MST Configuration Identifier for the switches within the MST Region `REGION567` in the testing topology.

5.3 Link Failure Scenario

In this section we will look into the behavior of the implemented solution after a link failure and reestablishment. For this types of scenarios, there is a `ScenarionManager` module within the INET framework. It serves for setting up and controlling the network simulations in OMNeT++.

The input parameter for the `ScenarioManager` module is a XML `script` parameter. To test the mentioned scenarios, we will use the following commands within the script:

- `<at t='30s'>` – used for grouping more commands to be executed at the same simulation time `t`,
- `<disconnect src-module='switch6' src-gate='ethg[0]'>` – disconnects a module from a link, if the specified source gate is bidirectional, both directions are disconnected,
- `<connect src-module='switch6' src-gate='ethg[0]' dest-module='switch1' dest-gate='ethg[3] channel-type='inet.node.ethernet.Eth10M'>` – we will use this commands to reconnect two modules, all of the parameters are required.

5.3.1 Link Failure

When the link failure/disconnect occurs, the `Mstp` module may notice the change on the link – the `NetworkInterface` (interface entry for the interface table in the `IInterfaceTable` module) by subscribing to the signal `interfaceStateChangedSignal`. Then, by implementing the `ReceiveSingnal` method within the module, we may configure the behavior on the switches on which the signal is received.

In current state, the created solution is slightly different from the referential behavior in this topology. This is caused because the Cisco has implemented the detection of unidirectional link failure, which is not present in any IEEE MSTP Standard [19]. The unidirectional link failure detection improvement checks the consistency of the port roles and

states within the received BPDU packets. It is made to prevent bridging loops, which may be caused by this type of link failure. This protocol improvements are not implemented.

In the testing topology, the link disconnection was simulated at time 30s. The example of monitored changes for the region `REGION567` is shown in Figure 5.12 and 5.13. Comparison was made with the captured packets in Wireshark on individual interfaces, an example displayed in Figure 5.14. The comparison is however irrelevant because of improved implementation of the MSTP behavior on Cisco switches after the already mentioned unidirectional link failure detection.

Event#	Time	Relevant Hops	ID / Source	Info / Destination		
#4095	30.000	switch5 --> switch6	BPDU AA-AA-AA-00-00-05	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4096	30.000	switch5 --> switch7	BPDU AA-AA-AA-00-00-05	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4098	30.000	switch6 --> switch5	BPDU AA-AA-AA-00-00-06	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4099	30.000	switch6 --> switch7	BPDU AA-AA-AA-00-00-06	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4100	30.000	switch7 --> switch5	BPDU AA-AA-AA-00-00-07	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4101	30.000	switch7 --> switch6	BPDU AA-AA-AA-00-00-07	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4211	30.000'124	switch5 --> switch6	BPDU AA-AA-AA-00-00-05	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4212	30.000'124	switch5 --> switch7	BPDU AA-AA-AA-00-00-05	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4217	30.000'124	switch7 --> switch6	BPDU AA-AA-AA-00-00-07	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4285	30.000'238'450	switch6 --> switch5	BPDU AA-AA-AA-00-00-06	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4286	30.000'238'450	switch6 --> switch7	BPDU AA-AA-AA-00-00-06	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4287	30.000'238'450	switch7 --> switch5	BPDU AA-AA-AA-00-00-07	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)
#4322	30.000'248	switch7 --> switch6	BPDU AA-AA-AA-00-00-07	01-80-C2-00-00-00	(UNIMPLEMENTED STP)	(inet::MSTBpdu)

Figure 5.12: The message exchange between switches within the MST Region `REGION567` after the link disconnect in the testing topology.

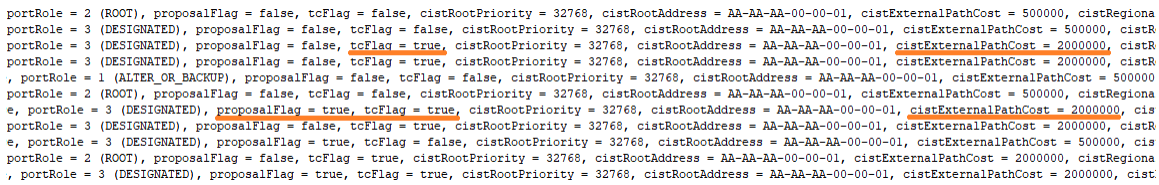

Figure 5.13: The message exchange between switches within the MST Region `REGION567` after link disconnect in the testing topology.

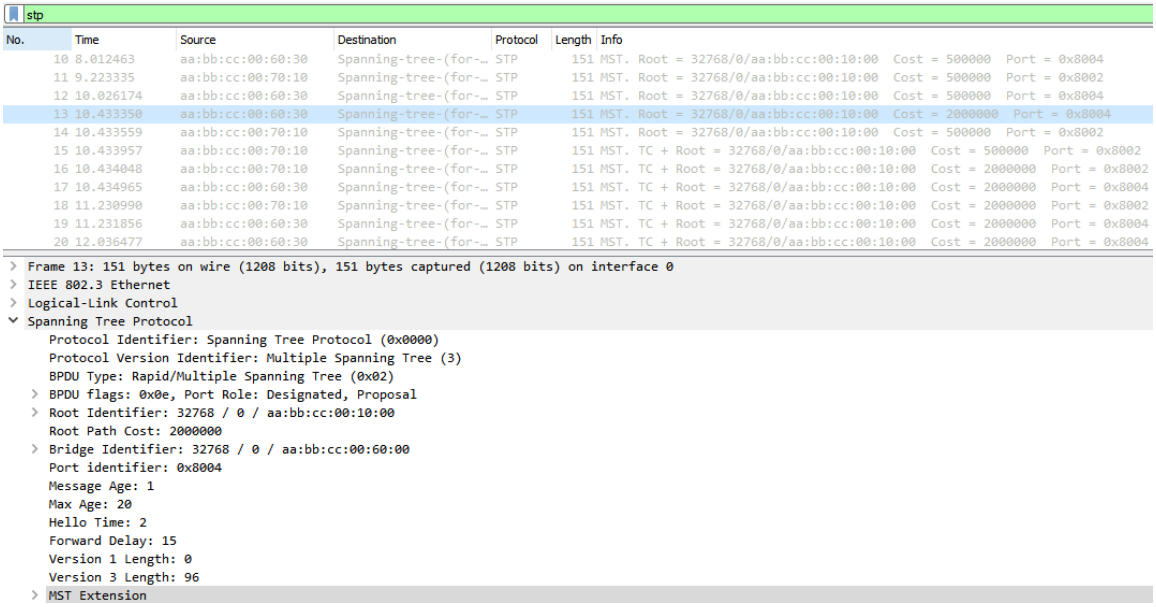

Figure 5.14: Packets received and sent from the Ethernet interface `eth0/3` on Cisco switch `S6` (the link between `S6` and `S7` switches).

5.3.2 Link Reestablishment

After the link is reconnected, the changes to the topology are reflected almost immediately thanks to the proposal-agreement mechanism. The changes may be monitored as well as in the section 5.3.1 within the transmitted packets.

The topology establishment before and after both, the link disconnect and reconnect is shown in Figure 5.15. The displayed changes are captured in simulation times 25s, 50s and 65s. The link disconnect was managed at simulation time 30s and the link reconnect at 60s.

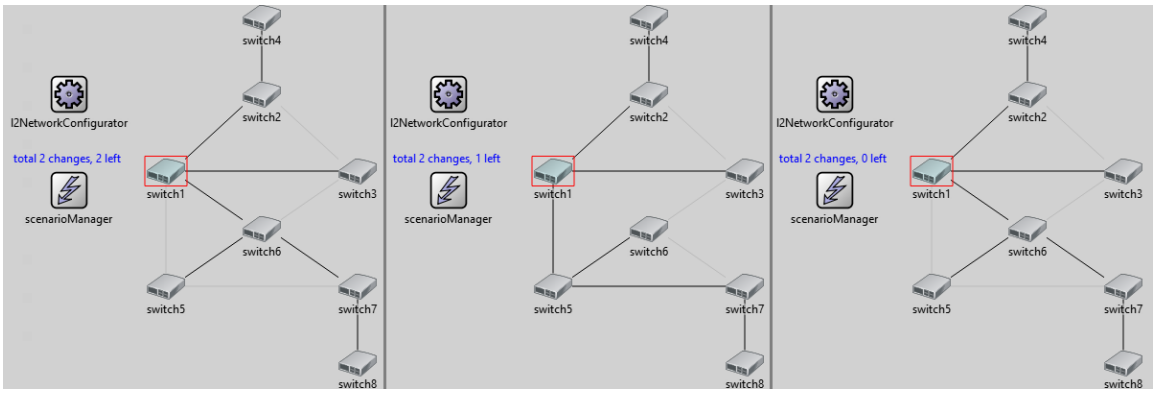

Figure 5.15: The established topology before and after the link disconnect and connect changes within `scenarioManager` in the simulated topology. The current tree for the CIST is highlighted.

5.4 Summary and Future Work

The implemented solution determines the MSTP topology according to the IEEE MSTP Standard. There is possibility to simulate a link failure scenarios. The GitHub repository, where the solution is beeing developed is located at this address[2].

There are still possible enhancements. The future work may include the reconnection of the implemented `Mstp` module with other modules in the INET framework. For example with module for managing VLANs on a link layer. Also the fallback to the previous versions of the Spanning Tree Protocols is not a part of implemented solution. I hope that in the near future, the implemented solution will be also a part of the INET framework.

[2] GitHub repository – https://github.com/simona5108/inet/tree/ieee8021s

Chapter 6

Conclusion

In this project, we discussed the functionality of STP variations, especially the Rapid Spanning Tree Protocol and the Multiple Spanning Tree Protocol. We described the characteristics of these protocols, including the description of the spanning tree creation for each protocol. The spanning tree protocols provide the opportunity to manage the active topology and to avoid loops with redundant links in the topology.

We also illustrated the possibilities of configuring and monitoring the spanning tree protocols on Cisco devices. We showed various configuration commands were for both, the RSTP, and the MSTP.

The main part of this work was the implementation of the MSTP into the INET framework within the OMNeT++ simulation library and framework used for building network simulators. The implementation was created based on the state machine definition in the IEEE802.1Q Standard (2018).

In order to use the solution, the user is to configure the MSTP parameters including port priorities and costs, bridge priorities, etc. Upon execution, the steps of the simulation can be monitored inside the OMNeT++ simulator. For an easy recognition of current state, we used the WATCH macros provided by OMNeT++.

We tested and compared the implementation with the referential behavior of the protocol on Cisco switches. The way of dealing with a link failure is not identical. The Cisco implementation contains the detection of unidirectional link failure which is not part of any IEEE MSTP Standard. These improvements are not contained within the current implementation.

During future development, the solution might become a part of the official INET framework. Nevertheless, it provides a valuable contribution to all INET framework users. It enables simple simulation of the MSTP. Further development might include binding the implemented module with other modules within the link layer, especially the modules for managing the VLANs. Additionally, the fallback to the previous versions of STP can be developed.

Bibliography

[1] *A Quick Overview of the OMNeT++ IDE* [online]. 2020 [cit. 2021-01-10]. Available at: https://omnetpp.org/documentation/ide-overview/.

[2] *Cisco Embedded Service 2020 Series Software Configuration Guide Cisco IOS Release 15.0(2)EC* [online]. 2013 [cit. 2021-01-13]. Available at: https://www.cisco.com/c/en/us/td/docs/switches/lan/embedded/software/release/15_0_2_ec/configuration/guide/ess_2020_scg.html.

[3] *Cisco Nexus 5600 Series NX-OS Layer 2 Switching Configuration Guide, Release 7.x* [online]. 2020 [cit. 2021-01-12]. Available at: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5600/sw/layer2/7x/b_5600_Layer2_Config_7x.html.

[4] *Configure the Transmit Hold Count - Sun Ethernet Fabric Operating System* [online]. 2012 [cit. 2021-05-02]. Available at: https://docs.oracle.com/cd/E39109_01/html/E21706/z4006bc41437804.html.

[5] *Configuring MST instances* [online]. 2015 [cit. 2021-01-21]. Available at: https://techhub.hpe.com/eginfolib/networking/docs/switches/WB/15-18/5998-8156_wb_2926_atmg/content/ch03s07.html.

[6] Hart, C. *Deep Dive - IEEE 802.1D Spanning Tree Topology Change – Christopher Hart - IT Adventures – Documenting my discoveries in the IT world* [online]. 2019 [cit. 2020-12-09]. Available at: https://www.chrisjhart.com/Deep-Dive-Spanning-Tree-Topology-Change/.

[7] *EtherSwitch* [online]. 2020 [cit. 2021-01-18]. Available at: https://doc.omnetpp.org/inet/api-current/neddoc/inet.node.ethernet.EtherSwitch.html.

[8] *HP A-Series Switches - Spanning-Tree 802.1D Bridge Protocol Data Unit (BPDU) Format* [online]. 2020 [cit. 2020-11-22]. Available at: https://support.hpe.com/hpesc/public/docDisplay?docId=mmr_kc-0124074.

[9] Hucaby, D. *CCNP Routing and Switching SWITCH 300-115 Official Cert Guide.* 1st ed. Cisco Press, 2015. 560 p. ISBN 978-1-58720-560-6.

[10] Richardson, S. *Identifying the Rstp Tcn Process - Switched Networks* [online]. 2020 [cit. 2020-12-21]. Available at: https://www.ccexpert.us/switched-networks/identifying-the-rstp-tcn-process.html.

[11] IEEE Standard for Information Technology -Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Common Specifications - Part 3: Media Access Control (MAC) Bridges: Amendment 2 - Rapid

Reconfiguration. *IEEE Std 802.1w-2001*. 2001, p. 1–116. DOI: 10.1109/IEEESTD.2001.93287.

[12] IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks. *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*. 2018, p. 1–1993. DOI: 10.1109/IEEESTD.2018.8403927.

[13] IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks. *IEEE Std 802.1Q-2005 (Incorporates IEEE Std 802.1Q1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002)*. 2006, p. 1–300. DOI: 10.1109/IEEESTD.2006.216285.

[14] IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks. *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*. 2014, p. 1–1832. DOI: 10.1109/IEEESTD.2014.6991462.

[15] IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges. *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*. 2004, p. 1–281. DOI: 10.1109/IEEESTD.2004.94569.

[16] IEEE Standard for Local Area Network MAC (Media Access Control) Bridges. *ANSI/IEEE Std 802.1D, 1998 Edition*. 1998, p. 1–373. DOI: 10.1109/IEEESTD.1998.95619.

[17] IEEE Standards for Local and Metropolitan Area Networks - Amendment to 802.1Q Virtual Bridged Local Area Networks: Multiple Spanning Trees. *IEEE Std 802.1s-2002 (Amendment to IEEE Std 802.1Q, 1998 Edition)*. 2002, p. 1–211. DOI: 10.1109/IEEESTD.2002.94223.

[18] Kraus, Z. *Modelování a analýza spolehlivosti počítačové sítě VUT*. Brno, CZ, 2011. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Available at: https://www.fit.vut.cz/study/thesis/9468/.

[19] *Layer 2/3 - Configuring Multiple Spanning-Tree Protocol [Cisco Catalyst 3850 Series Switches] - Cisco Systems* [online]. 2013 [cit. 2021-05-17]. Available at: https://www.cisco.com/en/US/docs/switches/lan/catalyst3850/software/release/3se/consolidated_guide/b_consolidated_3850_3se_cg_chapter_01001010.html#ID173.

[20] *Mac_address-table_aging-time.html - Cisco* [online]. 2012 [cit. 2021-01-17]. Available at: https://www.cisco.com/c/m/en_us/techdoc/dc/reference/cli/nxos/commands/l2/mac-address-table-aging-time.html.

[21] *MST BPDUs - NE20E-S V800R010C10SPC500 Feature Description - LAN Access and MAN Access 01 - Huawei* [online]. 2019 [cit. 2021-01-13]. Available at: https://support.huawei.com/enterprise/en/doc/EDOC1100055122/9ad92e66/mst-bpdus.

[22] *RSTP Port States - BCMSN - Cisco Certified Expert* [online]. 2020 [cit. 2020-11-26]. Available at: https://www.ccexpert.us/bcmsn/rstp-port-states.html.

[23] *Spanning-Tree Protocol Overview - TechLibrary - Juniper Networks* [online]. 2020 [cit. 2020-11-17]. Available at: https://www.juniper.net/documentation/en_US/junos/topics/topic-map/spanning-tree-overview.html.

[24] *Spanning Tree Protocol Principle - Programmer Sought* [online]. 2021 [cit. 2021-05-12]. Available at: https://programmersought.com/article/45151714417/#%E4%B8%89mstp.

[25] *Spanning Tree Protocol (STP) Overview - Cisco Meraki* [online]. 2020 [cit. 2020-11-03]. Available at: https://documentation.meraki.com/MS/Port_and_VLAN_Configuration/Spanning_Tree_Protocol_(STP)_Overview.

[26] *STP Feature Overview and Configuration Guide - stp_feature_config_guide.pdf* [online]. 2018 [cit. 2020-11-19]. Available at: https://www.alliedtelesis.com/sites/default/files/documents/feature-guides/stp_feature_config_guide.pdf.

[27] *STP versions and Port states – TutorZine* [online]. 2020 [cit. 2020-11-18]. Available at: https://tutorzine.com/stp-versions-stp-port-states/.

[28] *Understanding Multiple Spanning Tree Protocol (802.1s) - Cisco* [online]. 2007 [cit. 2021-01-13]. Available at: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24248-147.html#mst_region.

[29] *Understanding Rapid Spanning Tree Protocol (802.1w) - Cisco* [online]. 2017 [cit. 2020-12-13]. Available at: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html#anc16.

[30] *Understanding Spanning Tree Protocol Timers - Huawei* [online]. 2020 [cit. 2020-11-25]. Available at: https://support.huawei.com/enterprise/en/doc/EDOC1100092143.

[31] Wehrle, K., Günes, M. and Gross, J. *Modeling and Tools for Network Simulation*. 1st ed. Springer Science & Business Media, 2010. 545 p. ISBN 978-3-642-12331-3.

Appendix A

Contents of the included storage media

The enclosed DVD content is following:

- this document in PDF format (xpolac27.pdf)
- „readme.txt“ manual describing the project compilation
- directory „tex“ with latex source of this document
- directory „src“ containing the source code of the project implementation
- directory „captures“ including the captured files for analyzing the testing topology
- directory „install“ containing the files for OMNeT++ installation

Appendix B

Port Role Transitions SM Generalization

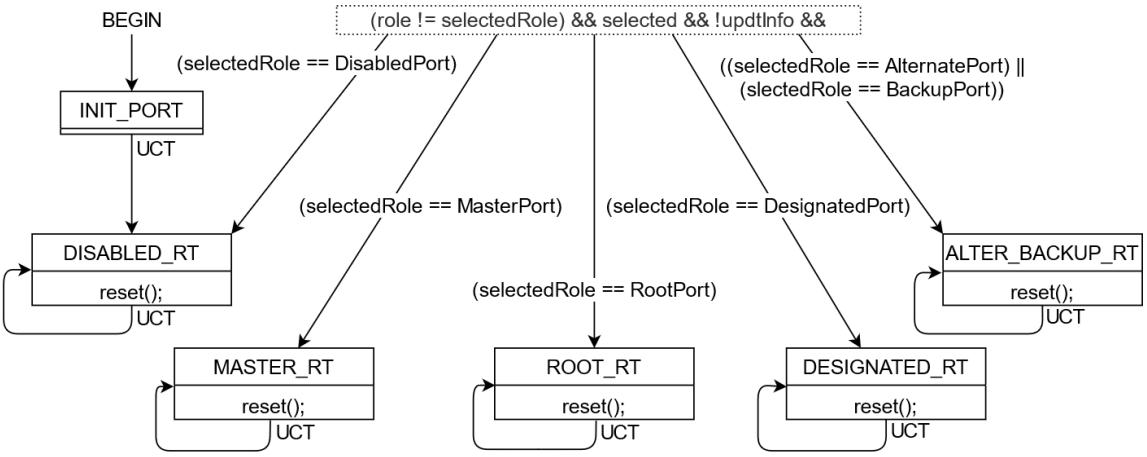

Figure B.1: Generalized Port Role Transitions SM.

The implementation of MSTP is based on the state machines from the IEEE 802.1Q Standard (2018). One of them is referred as the Port Role Transitions State Machine, which is constituted by: the Port Role Transitions SM for Disabled Port, Master Port, Root Port, Designated Port and one SM for both, Alternate and Backup Port. The connection of these state machines into one SM is not a part of the standard.

However, the SM displayed in Figure B.1 is part of the implemented solution. The input condition to each Port Role Transitions SM from the standard is generalized into one condition. The corresponding SM is then chosen based on the `selectedRole` parameter. The added reset method replaces just the input condition for individual Port Role Transitions state machines.

Appendix C

XML DTD

The following figures represent the document type definitions for the program configurations.

```
<?xml version="1.0"?>
<!DOCTYPE config [
    <!ELEMENT config        (interface*)>
    <!ELEMENT interface     (instance*)>

    <!ATTLIST interface
              hosts         CDATA   #REQUIRED
              ports         CDATA   #REQUIRED
              cost          CDATA   #IMPLIED
              priority      CDATA   #IMPLIED>
    <!ATTLIST instance
              id            CDATA   #REQUIRED
              cost          CDATA   #IMPLIED
              priority      CDATA   #IMPLIED>
]>
```

Figure C.1: XML DTD for Interface Configuration.

```
<?xml version="1.0"?>
<!DOCTYPE mstConfig [
    <!ELEMENT mstConfig     (instance*)>

    <!ATTLIST mstConfig
              name          CDATA   #IMPLIED
              revision      CDATA   #IMPLIED>
    <!ATTLIST instance
              id            CDATA   #REQUIRED
              vlan          CDATA   #IMPLIED>
]>
```

Figure C.2: XML DTD for Configuration of Instance to VLAN Mapping.

```
<?xml version="1.0"?>
<!DOCTYPE mstiBPConfig [
    <!ELEMENT mstiBPConfig  (instance*)>

    <!ATTLIST instance
              id            CDATA   #REQUIRED
              priority      CDATA   #IMPLIED>
]>
```

Figure C.3: XML DTD for MSTI Bridge Priority Configuration.